Learning to Truncate Ranked Lists for Information Retrieval

Chen Wu, Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
{wuchen17z, zhangruqing, guojiafeng, fanyixing, lanyanyan, cxq}@ict.ac.cn

Abstract

Ranked list truncation is of critical importance in a variety of professional information retrieval applications such as patent search or legal search. The goal is to dynamically determine the number of returned documents according to some user-defined objectives, in order to reach a balance between the overall utility of the results and user efforts. Existing methods formulate this task as a sequential decision problem and take some pre-defined loss as a proxy objective, which suffers from the limitation of local decision and non-direct optimization. In this work, we propose a global decision based truncation model named AttnCut, which directly optimizes user-defined objectives for the ranked list truncation. Specifically, we take the successful transformer architecture to capture the global dependency within the ranked list for truncation decision, and employ the reward augmented maximum likelihood (RAML) for direct optimization. We consider two types of user-defined objectives which are of practical usage. One is the widely adopted metric such as F1 which acts as a balanced objective, and the other is the best F1 under some minimal recall constraint which represents a typical objective in professional search. Empirical results over the Robust04 and MQ2007 datasets demonstrate the effectiveness of our approach as compared with the state-of-the-art baselines.

Introduction

Existing information retrieval (IR) systems mainly focus on relevance ranking which returns a ranked list of documents according to their relevance scores. Recently, ranked list truncation has attracted much attention in the IR community (Arampatzis, Kamps, and Robertson 2009; Lien, Cohen, and Croft 2019; Culpepper, Diaz, and Smucker 2018). Generally, the task aims to dynamically determine the number of returned documents according to some user-defined objectives, so as to reach a balance between the overall relevance or utility of the returned results and user efforts. Such truncation task is of critical importance in a variety of professional IR applications where user efforts could not be neglected. For example, in patent search (Lupu, Hanbury et al. 2013), it is time-consuming for a user to investigate each returned patent. In paid legal search (Tomlinson et al. 2007), litigation support professionals are paid by hour, thus each additional returned document would lead to some monetary penalty.

Without loss of generality, there are two typical truncation requirements in practical IR applications. Firstly, the truncation needs to reach a balance between the precision and recall of the returned results, leading to the optimization of a mixed metric of the two, e.g., the F1 score. In other words, the system needs to automatically determine the cut-off position by predicting the best F1 score. Secondly, in some scenario, recall is very critical which needs to pay more attention. For example, in patent search, users often require the returned list of patents to reach a target recall as they want to find whether there exist conflict patents. In such scenario, the system needs to determine the cut-off position with respect to the target metric such as F1, under some minimal recall constraint.

The present state-of-the-art method for ranked list truncation is BiCut (Lien, Cohen, and Croft 2019). BiCut takes this problem as a sequential decision process and adopts a Bi-directional Long Short-Term Memory (Bi-LSTM) (Graves 2013) model to solve it. Specifically, given a ranked list of documents with relevance score and document statistical information, BiCut attempts to predict Continue or EOL (end of the list) at each rank position, and decides to cut the ranked list at the first occurrence of EOL. The model is learned towards some pre-defined loss as a proxy objective of some user-defined metric.

However, the BiCut model suffers from the following two drawbacks. Firstly, as the truncation problem is formulated as a sequential decision process, the final cut-off decision is thus made upon a sequence of local decisions which may not be optimal from a global view. Secondly, although the work claims to optimize arbitrary user-defined metrics, the actually relationship between the defined loss function and the true F1 metric is not clear.

To address these problems, in this paper, we propose a global decision based truncation model named AttnCut, which directly optimizes user-defined objectives for the ranked list truncation. Specifically, we take the successful transformer architecture to capture the long-range dependency within the ranked list. In this way, the truncation decision could be made in a global way using the self-attention mechanism. Meanwhile, we employ the reward augmented maximum likelihood (RAML) (Norouzi et al. 2016) for the

model learning, which can directly optimize the user-defined metric such as F1 and DCG. Besides direct optimization of the target metric, we also tackle the prediction task of the best metric score under some minimal recall constraint.

We conduct empirical experiments on two widely adopted ad-hoc retrieval datasets, including the Robust04 dataset and the MQ2007 dataset (Qin et al. 2010). For evaluation, we compare with several state-of-the-art methods to verify the effectiveness of our model. Empirical results demonstrate that our model can well determine the number of returned documents and outperform all the baselines significantly on the two datasets.

Related work

In this section, we briefly review two lines of related work, including the ranked list truncation and reward augmented maximum likelihood method.

Ranked List Truncation

The goal of the ranked list truncation is to determine a best truncation position according to the input ranked list. Existing methods on ranked list truncation can be generally categorized into parametric methods and assumption-free methods. Parametric methods assume a prior distribution and find the best truncation position by fitting it. Early work mainly focuses on modeling score distributions by fitting parametric probability distributions (Manmatha, Rath, and Feng 2001). Arampatzis (Arampatzis, Kamps, and Robertson 2009) finds the best cut-off value over ranked lists which optimizes the F1-measure. By making the assumption that the score distributions of query-document pairs are normal for relevant and exponential for non-relevant, they adopt the Expectation Maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) to estimate the parameters. However, this method is under the normal-exponential mixture score distribution assumption (Arampatzis et al. 2001; Arampatzis 2002; Arampatzis and van Hameran 2001) which does not always hold.

Assumption-free approaches, on the other hand, aim to learn from the score distribution over the retrieval model using some machine learning methods and determine where to truncate (Wang, Lin, and Metzler 2011; Culpepper, Clarke, and Lin 2016; Lien, Cohen, and Croft 2019). Assumption-free approaches include Cascade-style approaches and some recent deep learning methods. Cascade-style approaches (Wang, Lin, and Metzler 2011) view retrieval as a multi-stage progressive refinement problem and decide the set of documents to prune at each stage, in order to achieve a balance between efficiency and effectiveness. In addition, Culpepper et al. (Culpepper, Clarke, and Lin 2016) use the score information over a set of sampled documents for learning dynamic cut-offs within cascade-style ranking systems. Deep learning methods apply deep architectures to do the truncation. For example, Bicut (Lien, Cohen, and Croft 2019) applies a Bi-LSTM model to take the sequential relations between documents' score and statistical information into consideration. They mention that any user-defined metric could be maximized if there is an appropriate corresponding loss function for minimization. However, the process of constructing such loss function is not declared. Besides, the weight hyper-parameters also increase the difficulty of application and explanation. Thus, in this work, we employ the RAML to directly optimize user-defined metric. Recently, Choppy (Bahri et al. 2020) takes the transformer architecture to truncate the ranked list. They optimize the metric by maximizing the expected evaluation metric on the training samples. While this method is heuristic, we are under the theoretical framework of RAML to make our model distribution approach the metric distribution and optimize user-defined metric directly and smoothly.

Reward Augmented Maximum Likelihood

Reward Augmented Maximum Likelihood (RAML) (Norouzi et al. 2016; Dai, Xie, and Hovy 2018) is a method which considers the task reward optimization over maximum likelihood estimation (MLE). By applying an exponentiated scale over the task reward and sample from it to get outputs, RAML optimizes log-likelihood on such output samples and corresponding inputs. If we consider the exponentiated pay-off distribution as the target distribution, RAML could be regarded as minimizing the KL divergence between the target distribution and the model distribution, while MLE is the same except that the target distribution is the Dirac distribution of ground-truth label. In this sense, by elaborating a different target distribution, RAML alleviates the exposure bias (Ranzato et al. 2015) as well as creates exploration opportunity for model to learn from sequences which are not exactly the same as the ground truth but have high rewards. On the other hand, compared with reinforcement learning (RL) algorithms such as policy gradient (Williams 1992) which optimizes task metric directly, RAML samples from a stationary reward distribution instead of the model distribution which is consistently changing. As a result, RAML avoids the high variance in gradient which is a well-known drawback of RL and enjoys a more stable optimization. We adopt RAML to directly optimize user-defined metric which could be regarded as a reward. Previous works have only applied RAML into image captioning, machine translation and sentence summarization (Dai, Xie, and Hovy 2018; Ma et al. 2017; Sperber, Niehues, and Waibel 2017; Li et al. 2018), to the best of our knowledge, AttnCut is the first model that applies RAML to the ranked list truncation.

Our Approach

In this section, we introduce the AttnCut model, a novel attention-based global decision model designed for the ranked list truncation task.

Model Overview

Formally, given a ranked document list $D = \{d_1, d_2, ..., d_N\}$ for a query q, AttnCut aims to find a best truncation position $k \in [1, N]$ that maximizes an external metric (Lien, Cohen, and Croft 2019).

Basically, our AttnCut model could be decomposed into three dependent components: 1) Encoding Layer: to obtain

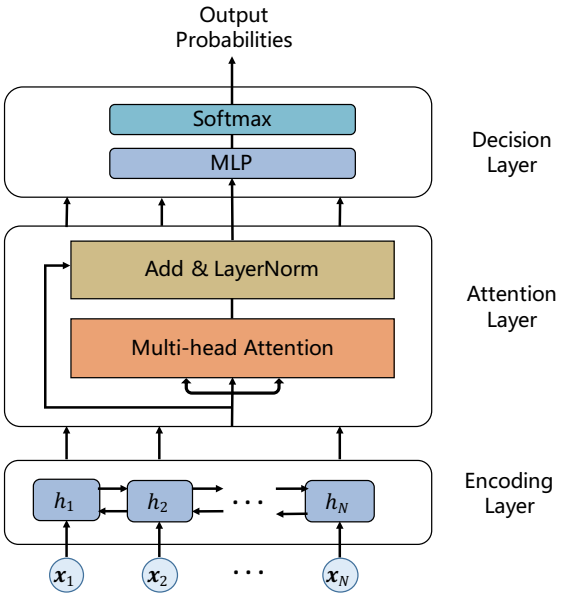

Figure 1: The overall architecture of AttnCut model

the representation of each document in the ranked list; 2) Attention Layer: to capture the long-range dependencies within the ranked document list through a direct connection between every pair of documents; 3) Decision Layer: to predict the final cut-off position based on the final representation of the ranked list. The overall architecture of BiCut is depicted in Figure 1, and we will detail our model as follows.

Encoding Layer

Generally, the encoding layer takes in the input ranked documents, and encodes them into a series of hidden representations.

Each document d_n in each ranked document list D is firstly represented by its feature vector $\mathbf{x}_n = [r_n||s_n]$, which is achieved by concatenating a relevance score r_n given by a retrieval function (e.g., BM25) and corresponding document statistic s_n, including document length, number of unique tokens, and document similarity. Specifically, we follow (Lien, Cohen, and Croft 2019) to compute the document length and the number of unique tokens. The document similarity denotes some pre-defined cosine similarity (e.g., tf-idf and doc2vec) between a document and its neighborhood documents. Then, we use a two-layer bi-directional LSTM as the document encoder, which summarizes not only the preceding documents, but also the following documents. The document encoder is used to sequentially receive the feature vectors of documents $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and the hidden representation $\mathbf{h}_n$ of each document $d_n \in D$ is given by concatenating the forward and backward hidden states of the second layer in the document encoder.

Attention Layer

The attention layer aims to capture long-term dependencies within the ranked document list. The key idea is that the final cut-off decision should be made upon a global way. To achieve this purpose, we leverage a single-layer transformer architecture (Vaswani et al. 2017) over the hidden representations $\mathbf{h}_n$ of each document d_n in the input ranked list. In particular, the multi-head attention mechanism in Transformer allows every document to be directly connected to any other documents in a ranked document list.

Specifically, in the multi-head attention mechanism, each document will attend to all the documents and obtains a set of attention scores that are used to refine its representation. Given current document representations $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_N\}$, refined new document representations $\mathbf{M}$ are calculated as:

$$\begin{aligned} \mathbf{M} &= \text{MultiHeadAttention}(\mathbf{H}) \\ &= \text{Concatenation}(\text{head}_1, ..., \text{head}_h)\mathbf{W_H}, \\ \text{head}_i &= \text{softmax}(\frac{(\mathbf{Q^{(i)}})(\mathbf{K^{(i)}})^T}{\sqrt{t}})(\mathbf{V^{(i)}}), \end{aligned} \tag{1}$$

where h is the number of heads and $\mathbf{Q^{(i)}} = \mathbf{H}\mathbf{W}_\mathbf{Q}^{(\mathbf{i})}, \mathbf{K^{(i)}} = \mathbf{H}\mathbf{W}_\mathbf{K}^{(\mathbf{i})}, \mathbf{V^{(i)}} = \mathbf{H}\mathbf{W}_\mathbf{V}^{(\mathbf{i})}$. $\mathbf{W_H} \in \mathbb{R}^{t\times t}$ and $\mathbf{W}_\mathbf{Q}^{(\mathbf{i})}$, $\mathbf{W}_\mathbf{K}^{(\mathbf{i})}$, $\mathbf{W}_\mathbf{V}^{(\mathbf{i})} \in \mathbb{R}^{t\times(t/h)}$ are learnable parameters with t as the model dimension. The dimension scaling factor $\frac{1}{\sqrt{t}}$ is applied to adapt the fast growth in dot-product attention.

After obtaining the refined representation of each document by the multi-head attention mechanism, we add a layer normalization (Ba, Kiros, and Hinton 2016) to obtain the final representation of the ranked document list $\mathbf{M}' \in \mathbb{R}^{N\times t}$,

$$\mathbf{M}' = \text{LayerNorm}(\mathbf{M} + \mathbf{H}). \tag{2}$$

Decision Layer

The goal of the decision layer is to identify an appropriate cut-off position k for each query q given the final representation of the input ranked list $\mathbf{M}'$. Specifically, we arrive at the output probability of AttnCut by applying a multilayer perceptron (MLP) followed by a softmax over positions in the ranked document list:

$$\boldsymbol{p} = \text{Softmax}(\text{MLP}(\mathbf{M}')), \tag{3}$$

where $\boldsymbol{p} = \{p_n\}_{n=1}^N \in \mathbb{R}^{N\times 1}$ stands for a probability distribution over candidate N cut-off positions.

Model Training and Inference

In the training phase, we propose to take into account the alternative outputs beyond the ground truth for better model learning, meanwhile attempt to keep the optimization procedure simple and efficient. The key idea is that if we can derive a better target distribution (i.e., user-defined metric) which can convey the information of the output structure, we can then directly use it to replace the Dirac distribution in the MLE objective to achieve our purpose.

Specifically, we try to derive the new target distribution by employing Reward Augmented Maximum Likelihood (RAML) (Norouzi et al. 2016), which consists of the following two steps.

- **Define Output Distribution**. Without loss of generality, given the ground-truth relevance label $\mathbf{y}^* = \{y_1^*, \ldots, y_N^*\}$ ($y_n^* = 1$ if d_n is relevant, $y_n^* = -1$ if d_n is non-relevant) of the ranked list D and a reward function r, we can compute the reward $r_k(\mathbf{y}^*)$ if the ranked list D is truncated at position k. Specifically, we take the user-defined metric (e.g., the evaluation metric $F1$ as defined at Equ.(10)) as the reward function. Following the idea in (Dai, Xie, and Hovy 2018), we normalize these rewards scores to obtain the distribution of the outputs as

$$q_k = \frac{\exp(r_k(\mathbf{y}^*)/\tau)}{\sum_{n=1}^{N} \exp(r_n(\mathbf{y}^*)/\tau)}, \tag{4}$$

  where τ is the hyper-parameter which controls the concentration of the distribution around $\mathbf{y}^*$. Obviously, this distribution reflects how the task rewards distributed in the output space.

- **Integrate into MLE Criterion**. Previous existing neural models rely on MLE for model learning. Specifically, the MLE criterion is to maximize the log-likelihood of the ground-truth truncation positions as follows:

$$\begin{aligned} \mathcal{L}_{MLE}(\theta) &= -\log p(D;\theta) \\ &= -\sum_k \delta_{k^*}(k) \log\ p_k(D;\theta), \end{aligned} \tag{5}$$

  where $p_k(D;\theta)$ is the output probability defined as Equ.(3), and δ_k denotes the Dirac distribution of the ground-truth truncation position, i.e., $\delta_{k^*}(k^*) = 1$ else $\delta_{k^*}(k) = 0$ for other k.

  As we can see, the MLE criterion ignores the structure of the output space by treating all the outputs that do not match the ground-truth as equally poor, and thus brings the discrepancy between training and test. Here, we replace the Dirac distribution δ_k of MLE in Equ.(5) with the above derived distribution q_k, and obtain our learning criterion as follows,

$$\begin{aligned} \mathcal{L}_r(\theta;\tau) &= -\sum_k q_k \log p_k(D;\theta) \\ &= -\mathbb{E}_{q_k}[\log\ p_k(D;\theta)]. \end{aligned} \tag{6}$$

This loss makes our model distribution approach the normalized reward distribution. Now we can directly optimize this new target distribution augmented objective function for learning AttnCut. We can see that this learning criterion is easy to implement in practice. It is also a general learning criterion to be adopted by almost all the existing ranked list truncation models.

In the inference phase, given a ranked document list $D = \{d_1, \ldots, d_N\}$ with respect to a query q, we pick the position k with the highest target metric as defined in Equ.(3) to cut the ranked list.

Recall-Constraint Model

In some scenarios, people have specific recall requirements to the ranked list. For example, in patent search, users often require the returned list of patents to reach a target recall as they want to find whether there exist conflict patents. Therefore, it is necessary for ranked list truncation model to decide the cut-off position with respect to the target metric under some minimal recall constraint. To achieve this purpose, we extend AttnCut to achieve an optimal target metric (e.g., F1) and ensure a target minimal recall simultaneously for the final cut-off decision.

Firstly, we compute the recall of the remaining ranked documents truncated at candidate cut-off positions $k \in [1, N]$, which is defined as,

$$R@k = \frac{1}{N_D} \sum_{n=1}^{k} \delta(y_n^* = 1), \tag{7}$$

where N_D denotes the number of relevant documents in the ranked list D and y_n^* is the relevance label of document d_n in the truncated ranked list. δ is the indicator function.

Then, we employ the same encoding layer and attention layer in AttnCut to obtain the final representation of the ranked list. Note we split $R@k \in [0, 1]$ into B ordered bins. Hence, we modify the decision layer to classify each candidate position into B ordered recall bins and achieve the probability distribution $\boldsymbol{p'} = \{p'_n\}_{n=1}^{N} \in \mathbb{R}^{N \times B}$. We learn the recall-constraint AttnCut using MLE objective since the B-dimension probability distribution of each position is not suitable as the reward score.

In the testing phase, we use AttnCut to pick the position m with the highest target metric (e.g., F1), and use recall-constraint AttnCut to pick the position j under the minimal recall requirement σ. If $m \geq j$, then m is the eligible position. Otherwise, a sub-optimal position m', i.e., a position with the sub-highest target metric, will be tried until $m' > j$.

Experiments

In this section, we conduct experiments to verify the effectiveness of our proposed model.

Dataset Description

We conduct experiments on two representative IR datasets.

- **Robust04** contains 250 queries and 528k news articles, whose topics are collected from TREC 2004 Robust Track[1]. There are about 70 relevant documents (news articles) for each query.
- **Million Query Track 2007 (MQ2007)** is a LETOR (Qin et al. 2010) benchmark dataset which uses Gov2 web collection. There are 1692 queries and 65323 documents, where each query has 10 relevant documents in average.

We leverage two widely adopted retrieve models, i.e., BM25 (Robertson and Walker 1994) and DRMM (Guo et al. 2016), to obtain the ranked list. Specifically, we retrieve the top 300 and top 150 documents as the ranked list for Robust04 and MQ2007, respectively. The detailed statistics of these datasets are shown in Table 1. Note we do not use the MS MARCO dataset (Bajaj et al. 2016) since most queries are associated with only one relevant document which is not suitable for ranked list truncation.

[1] https://trec.nist.gov/data/robust.html

	Robust04	MQ2007
#Queries	250	1692
#Documents	528k	65323
Query: # Ranked Documents	300	150
Query: avg #Relevant Documents	70	10

Table 1: Statistics of the two IR datasets.

Implementation Details

We implement our AttnCut model in PyTorch[2]. For two datasets, we randomly divide them into a training set (80%) and a testing set (20%) following (Lien, Cohen, and Croft 2019) to achieve comparable performance. For the Encoding Layer, we first compute the tf-idf and doc2vec of each document using gensim tool[3] over the whole corpus. The dimension of tf-idf and doc2vec is 648730 and 200 respectively. The LSTM hidden unit size of the two-layer bi-directional LSTM is set as 128. For the Attention Layer, the hidden size t of the Transformer is 256 and the number h of self-attention heads is 4. For training, the mini-batch size for the update is set as 20 and 128 for Robust04 and MQ2007 respectively. The parameter τ for RAML learning is set as 0.95. We apply stochastic gradient decent method Adam (Kingma and Ba 2014) to learn the model parameters with the learning rate as 3×10^{-5}. For recall-constraint model, we set the number of ordered bins B as 5.

Baselines

We adopt three types of baseline methods for comparison, including traditional truncation methods, neural truncation models and our model variants.

For traditional truncation methods, we apply three representative methods with different policies:

- **Oracle** uses the ground-truth labels of the test queries to find a best truncation position k for each query, which represents an upper-bound on the metric performance that can be achieved.
- **Fixed**-k determines a fixed point k across test queries to return the top k document results (Fan et al. 2018; Tay, Tuan, and Hui 2018; Wang and Nyberg 2015).
- **Greedy**-k chooses a fixed k over the training data to maximize the user-defined evaluation metric.

The neural truncation models include,

- **BiCut** (Lien, Cohen, and Croft 2019) proposes an RNN-based model combined with a flexible cost function, and predicts Continue and EOL for end-of-list. The ranked list is truncated at the first instance of EOL.
- **Choppy** (Bahri et al. 2020) leverages a Transformer architecture for ranked list truncation and optimizes the expected metric value.

Furthermore, we implement some variants of our model by using different learning objectives, including,

- **AttnCut-MLE** learns AttnCut by MLE as defined in Equ.5 to maximize the log-likelihood of the ground-truth truncation positions.
- **AttnCut-Bi** uses the loss function applied in Bicut (Lien, Cohen, and Croft 2019) to learn our AttnCut model, which is formalized as follows:

$$\begin{aligned}\mathcal{L}_{BiCut} = \sum_{d_n \in D_k} (\alpha \mathbb{I}(y_n^* = 0) \frac{p_n}{1-r} \\ + (1-\alpha)\mathbb{I}(y_n^* = 1)\frac{1-p_n}{r}),\end{aligned} \tag{8}$$

where y_n^* is the relevant label of the document d_n, $\mathbb{I}$ is an indicator function, r is the normalization factor and α is a hyper-parameter. D_k denotes the remaining ranked documents truncated at the cut-off position k.

- **AttnCut-RL** learns AttnCut by reinforcement learning (RL) (Williams 1992). We define the user-defined metric as a reward, and the RL objective function is defined as,

$$\mathcal{L}_{RL} = -\sum_k p_k(D;\theta)\gamma^{l-1} r_k(\mathbf{y}^*), \tag{9}$$

where $p_k(D;\theta)$ is the output probability defined as Equ.(3) and $\gamma^{l-1} r_k(\mathbf{y}^*)$ is the discounted reward. γ is the decay rate and l denotes for trace l in training.

Evaluation Metrics

As for evaluation measures, two standard evaluation metrics, i.e., F1 at rank k (F1@k) and discounted cumulative gain at rank k (DCG@k), are used in experiments following previous works (Lien, Cohen, and Croft 2019; Bahri et al. 2020).

- **F1@k** is evaluated at the cut-off candidate position k:

$$\begin{aligned} F_1@k &= \frac{2 * P@k * R@k}{P@k + R@k}, \\ P@k &= \frac{1}{k}\sum_{n=1}^{k} \delta(y_n^* = 1), \\ R@k &= \frac{1}{N_D}\sum_{n=1}^{k} \delta(y_n^* = 1), \end{aligned} \tag{10}$$

where $y_n^* \in \{-1, 1\}$ is the relevance label of the document d_n, and N_D denotes the number of relevant documents in the ranked list.

- **DCG@k** (Järvelin and Kekäläinen 2002) is also evaluated at the cut-off candidate position k:

$$DCG@k = \sum_{n=1}^{k} \frac{y_n^*}{\log_2(n+1)}. \tag{11}$$

For methods that optimize F1@k or DCG@k, we report the performance of the model when it is optimized specifically for that metric. Note that the widely used version of DCG (Burges et al. 2005) always increases monotonically with the list length k, leading to the best solution as no truncation. Here we adopt the definition of DCG from (Järvelin

[2]https://pytorch.org/

[3]https://radimrehurek.com/gensim/

Method	Robust04				MQ2007			
	BM25		DRMM		BM25		DRMM	
	F1@k	DCG@k	F1@k	DCG@k	F1@k	DCG@k	F1@k	DCG@k
AttnCut-MLE	0.2538	0.3338	0.2770	0.4416	0.3096	-0.0741	0.3536	-0.0241
AttnCut-Bi	0.2819	-	0.2870	-	0.3302	-	0.4008	-
AttnCut-RL	0.2733	0.3404	0.2808	0.6087	0.3248	-0.0716	0.3985	-0.0199
AttnCut	**0.2821**	**0.3846**	**0.2944**	**0.6496**	**0.3353**	**-0.0659**	**0.4047**	**-0.0144**

Table 2: Model analysis of our AttnCut using different learning objectives under the $F1@k$ and $DCG@k$.

Method	Robust04				MQ2007			
	BM25		DRMM		BM25		DRMM	
	F1@k	DCG@k	F1@k	DCG@k	F1@k	DCG@k	F1@k	DCG@k
Oracle	0.3591	1.3328	0.3863	1.5948	0.4767	1.0569	0.5570	1.5742
Fixed-k (5)	0.1550	0.1876	0.1601	0.3114	0.2175	-0.5966	0.2486	-0.3227
Fixed-k (10)	0.2103	-0.2672	0.2172	-0.1137	0.2794	-1.1860	0.3135	-0.8152
Fixed-k (50)	0.2499	-5.3966	0.2649	-4.9261	0.2704	-6.9066	0.3224	-4.1455
Greedy-k	0.2538	0.2245	0.2642	0.4580	0.3208	-0.0828	0.3965	-0.0592
BiCut	0.2513	-	0.2697	-	0.3231	-	0.3958	-
Choppy	0.2738	0.3631	0.2914	0.6269	0.3238	-0.0732	0.4011	-0.0368
AttnCut	**0.2821**†	**0.3846**	**0.2944**†	**0.6496**	**0.3353**†	**-0.0659**	**0.4047**†	**-0.0144**

Table 3: Comparisons between our AttnCut model and baselines for Robust04 and MQ2007 datasets. † represents statistical significance against BiCut model ($p < 0.05$, Wilcoxon test).

and Kekäläinen 2002) to penalize irrelevant documents since we have set $y_n^* = 1$ for relevant document and -1 for irrelevant. This monotony is also the reason why other commonly used ranking metrics such as MAP (Sanderson 2010) and MRR (Voorhees 1999) cannot be used in the truncation task.

For the evaluation of recall-constraint AttnCut, we also compute the recall defined in Equ.7 at candidate cut-off positions to verify whether the truncation results are under the recall constraint.

Evaluation Results

Model Analysis We first analyze the three types of AttnCut models to investigate which learning objetive is better for ranked list truncation. As shown in Table 2, we can find that: (1) *AttnCut-MLE* cannot work well. This is mainly because the MLE learning criterion brings the discrepancy between training and test, leading to overfitting on the ground-truth labels and reduced generalization ability. (2) *AttnCut-Bi* can achieve better results than *AttnCut-RL*, indicating that leveraging a joint loss function which controls the impact of false positives and false negatives is better than reinforcement learning with the target metric as the reward. (3) *AttnCut* achieves the best performance for two datasets as evaluated by all the metrics.

Baseline Comparison The performance comparisons between our model and the baselines are shown in Table 3. We can observe that: (1) For traditional truncation methods, the *fixed-k* methods perform poorly, indicating that simply returning the top-k results is not suitable for ranked list truncation. (2) *Fixed-k* on the fixed point 50 achieves the comparable results with *Greedy-k* on the Robust04 dataset in terms of F1. However, the best fixed point may vary across different datasets and evaluation metrics, resulting in the limitation of flexibility in truncating different ranked lists. Note that the comparative results between *Fixed-k* and *Greedy-k* are slightly different with that reported in (Lien, Cohen, and Croft 2019). The reason is that we split the datasets into the training and testing sets with different random seeds. (3) The neural truncation methods (i.e., *BiCut* and *Choppy*) can achieve better results than the traditional truncation methods, since these neural methods apply deep architectures to learn from the score distribution and truncate dynamically. (4) By learning the conditional joint distribution over candidate cut positions that maximizes the expected evaluation metric on the training samples, *Choppy* is able to achieve the best performance among all the baseline methods. However, compared with *Oracle*, there is still a large gap between *Choppy* and the upper-bound. (5) The better results of *AttnCut* over *BiCut* demonstrate the effectiveness of directly optimizing user-defined objectives, which captures the long-range dependency within the ranked list. (6) *AttnCut* outperforms *Choppy*, demonstrating the effectiveness of RAML which makes our model distribution approach the metric distribution is a better learning objective than that maximizes the expected evaluation metric. (7) *AttnCut* achieves the best

Recall Threshold	Robust04				MQ2007			
	BM25		DRMM		BM25		DRMM	
	F1@k	R@k	F1@k	R@k	F1@k	R@k	F1@k	R@k
$\sigma = 0$, Oracle	0.3591	0.3777	0.3863	0.4352	0.4767	0.6356	0.5570	0.7422
$\sigma = 0$, AttnCut	0.2821	0.3527	0.2881	0.3674	0.3059	0.4125	0.3959	0.7267
$\sigma = 0.3$, Oracle	0.3696	0.4641	0.3963	0.4948	0.5652	0.7631	0.6441	0.8617
$\sigma = 0.3$, AttnCut	0.2417	0.5726	0.2836	0.6558	0.4299	0.6442	0.4614	0.9049
$\sigma = 0.5$, Oracle	0.3600	0.5818	0.3967	0.5923	0.5688	0.8048	0.6412	0.8784
$\sigma = 0.5$, AttnCut	0.2171	0.6837	0.2522	0.7870	0.4453	0.7668	0.4612	0.9124
$\sigma = 0.7$, Oracle	0.3478	0.7470	0.3645	0.7467	0.5438	0.8935	0.6175	0.9361
$\sigma = 0.7$, AttnCut	0.1572	0.8323	0.1994	0.8794	0.4263	0.9074	0.4645	0.9467

Table 4: Performance of the extended recall-constraint model on Robust04 dataset and MQ2007 dataset. σ denotes the minimal recall threshold.

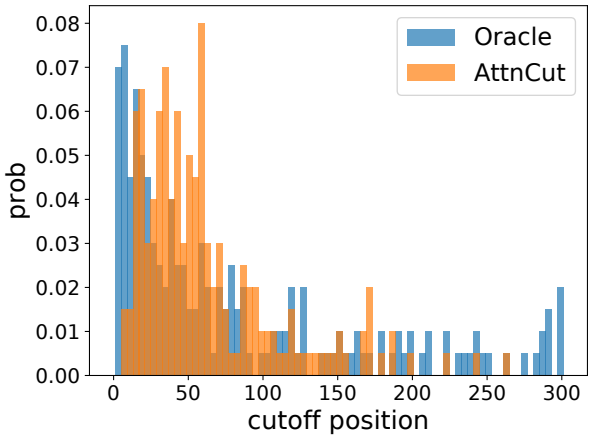

Figure 2: The distribution of cut-off positions of testing queries from AttnCut and Oracle.

performance. For example, for Robust04, the relative improvement of *AttnCut* against the *BiCut* is about 12.26% in terms of F1 under BM25 retrieve model.

Cut-off Position Distribution Analysis To better understand what can be learned in AttnCut, we conduct qualitative analysis on the distribution of cut-off positions of testing queries by comparing with the best cut-off given by Oracle. Specifically, we visualize the cut-off position distributions of AttnCut and Oracle over the ranked list retrieved by BM25 on Robust04 dataset in Figure 2 to help analysis. As we can see, AttnCut is able to approximate the optimal distribution of cut-off value, and truncates the ranked list well before the 300 document limit. The best truncation positions of AttnCut fall into the range of $[0, 50]$. The reason might be that most of relevant documents are ranked in top 50 by BM25. However, the inability to properly produce the optimal cut-off position when the position is greater than 250 suggests that the model learns a conservative approach to the truncation task (Lien, Cohen, and Croft 2019).

Recall-Constraint Results For the evaluation of our recall-constraint AttnCut model under different minimal recall requirements, we conduct a simulation experiment on Robust04 and MQ2007, and vary the user-given recall σ by setting it to four different values (i.e., 0, 0.3, 0.5, 0.7). Since there are no existing works on this task, we take a brute-force search over the ranked list to get the upper-bound on the metric performance as a comparison, which is denoted as *Oracle* in Table 4. To reveal whether the recalls of the truncated ranked list satisfy the specific recall requirements, we also show R@k value defined in Equ.7. Note *AttnCut* might outperform *Oracle* in terms of R@k since both *Oracle* and *AttnCut* are optimized towards F_1 with recall as a constraint.

As shown in Table 4, we can observe that: (1) The F_1 score of *Oracle* for Robust04 is much smaller than that for MQ2007, and the F_1 of *AttnCut* drops significantly on Robust04 with the minimal recall requirement increasing. The reason might be that the query in Robust04 is associated with more relevant documents than that in MQ2007 (i.e., 70 vs 10) and the recall value may be smaller (e.g., the R@k on Robust04 is worse than that on MQ2007). (2) Recall-constraint AttnCut could satisfy the recall requirements, demonstrating the effectiveness of determining the cut-off position w.r.t. the target metric under some minimal recall constraint.

Conclusion and Future Work

In this paper, we proposed to directly optimize user-defined objectives for the ranked list truncation, which aims to make the final cut-off decision from a global view. We leveraged the successful transformer architecture to capture the long-range dependency within the ranked list, and employed RAML for the model learning. Thus, the user-defined metric, which can convey the information of the output structure, could be directly optimized. Furthermore, we tackled the prediction task of the best target metric under some minimal recall constraint. Empirical results showed that our model can significantly outperform the state-of-the-art methods. In the future work, we would like to consider some diversity related document features to obtain better document representations.

Acknowledgments

This work was supported by Beijing Academy of Artificial Intelligence (BAAI) under Grants No. BAAI2019ZD0306 and BAAI2020ZJ0303, and funded by the National Natural Science Foundation of China (NSFC) under Grants No. 61722211, 61773362, 62006218, 61872338, and 61902381, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, the National Key RD Program of China under Grants No. 2016QY02D0405, the Lenovo-CAS Joint Lab Youth Scientist Project, the K.C.Wong Education Foundation, and the Foundation and Frontier Research Key Program of Chongqing Science and Technology Commission (No. cstc2017jcyjBX0059).

References

Arampatzis, A. 2002. Unbiased sd threshold optimization, initial query degradation, decay, and incrementality, for adaptive document filtering. *NIST SPECIAL PUBLICATION SP* (250): 596–603.

Arampatzis, A.; Beney, J.; Koster, C. H.; and van der Weide, T. P. 2001. Incrementality, half-life, and threshold optimization for adaptive document filtering. *Proc. of the 9th Text Retrieval Conference (TREC-9)* .

Arampatzis, A.; Kamps, J.; and Robertson, S. 2009. Where to stop reading a ranked list? Threshold optimization using truncated score distributions. In *SIGIR*, 524–531.

Arampatzis, A.; and van Hameran, A. 2001. The score-distributional threshold optimization for adaptive binary classification tasks. In *SIGIR*, 285–293.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv:1607.06450* .

Bahri, D.; Tay, Y.; Zheng, C.; Metzler, D.; and Tomkins, A. 2020. Choppy: Cut Transformer For Ranked List Truncation. *arXiv:2004.13012* .

Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv:1611.09268* .

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, 89–96.

Culpepper, J. S.; Clarke, C. L.; and Lin, J. 2016. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proceedings of the 21st Australasian Document Computing Symposium*, 17–24.

Culpepper, J. S.; Diaz, F.; and Smucker, M. D. 2018. Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in lorne (swirl 2018). In *ACM SIGIR Forum*, 34–90.

Dai, Z.; Xie, Q.; and Hovy, E. 2018. From credit assignment to entropy regularization: Two new algorithms for neural sequence prediction. *arXiv:1804.10974* .

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* 39(1): 1–22.

Fan, Y.; Guo, J.; Lan, Y.; Xu, J.; Zhai, C.; and Cheng, X. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR*, 375–384.

Graves, A. 2013. Generating sequences with recurrent neural networks. CoRR abs/1308.0850 (2013). *arXiv:1308.0850* .

Guo, J.; Fan, Y.; Ai, Q.; and Croft, W. B. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*.

Järvelin, K.; and Kekäläinen, J. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20(4): 422–446.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* .

Li, H.; Zhu, J.; Zhang, J.; and Zong, C. 2018. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *COLING*, 1430–1441.

Lien, Y.-C.; Cohen, D.; and Croft, W. B. 2019. An Assumption-Free Approach to the Dynamic Truncation of Ranked Lists. In *ICTIR*, 79–82.

Lupu, M.; Hanbury, A.; et al. 2013. Patent retrieval. *Foundations and Trends® in Information Retrieval* 7(1): 1–97.

Ma, X.; Yin, P.; Liu, J.; Neubig, G.; and Hovy, E. 2017. Softmax q-distribution estimation for structured prediction: A theoretical interpretation for raml. *arXiv:1705.07136* .

Manmatha, R.; Rath, T.; and Feng, F. 2001. Modeling score distributions for combining the outputs of search engines. In *SIGIR*, 267–275.

Norouzi, M.; Bengio, S.; Jaitly, N.; Schuster, M.; Wu, Y.; Schuurmans, D.; et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, 1723–1731.

Qin, T.; Liu, T.-Y.; Xu, J.; and Li, H. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13(4): 346–374.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv:1511.06732* .

Robertson, S. E.; and Walker, S. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR94*, 232–241. Springer.

Sanderson, M. 2010. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. ISBN-13 978-0-521-86571-5, xxi+ 482 pages. *Natural Language Engineering* 16(1): 100–103.

Sperber, M.; Niehues, J.; and Waibel, A. 2017. Toward robust neural machine translation for noisy input sequences. In *IWSLT*.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. Cross temporal recurrent networks for ranking question answer pairs. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tomlinson, S.; Oard, D. W.; Baron, J. R.; and Thompson, P. 2007. Overview of the TREC 2007 Legal Track. Citeseer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Voorhees, E. 1999. Proceedings of the 8th text retrieval conference. *TREC-8 Question Answering Track Report* 77–82.

Wang, D.; and Nyberg, E. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, 707–712.

Wang, L.; Lin, J.; and Metzler, D. 2011. A cascade ranking model for efficient ranked retrieval. In *SIGIR*, 105–114.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.