

Learning to Rank Words: Optimizing Ranking Metrics for Word Spotting

Pau Riba^[0000-0002-4710-0864], Adrià Molina^[0000-0003-0167-8756], Lluís
Gomez^[0000-0003-1408-9803], Oriol Ramos-Terrades^[0000-0002-3333-8812], and
Josep Lladós^[0000-0002-4533-4739]

Computer Vision Center and Computer Science Department,
Universitat Autònoma de Barcelona, Catalunya
{priba,lgomez,oriolrt,josep}@cvc.uab.cat,
adria.molinar@e-campus.uab.cat

Abstract. In this paper, we explore and evaluate the use of ranking-based objective functions for learning simultaneously a word string and a word image encoder. We consider retrieval frameworks in which the user expects a retrieval list ranked according to a defined relevance score. In the context of a word spotting problem, the relevance score has been set according to the string edit distance from the query string. We experimentally demonstrate the competitive performance of the proposed model on query-by-string word spotting for both, handwritten and real scene word images. We also provide the results for query-by-example word spotting, although it is not the main focus of this work.

Keywords: Word Spotting · Smooth-nDCG · Smooth-AP · Ranking Loss.

1 Introduction

Word spotting, also known as keyword spotting, was introduced in the late 90's in the seminal papers of Manmatha *et al.* [19,20]. It emerged quickly as a highly effective alternative to text recognition techniques in those scenarios with scarce data availability or huge style variability, where a strategy based on full transcription is still far from being feasible and its objective is to obtain a ranked list of word images that are relevant to a user's query. Word spotting has been typically classified in two particular settings according to the target database gallery. On the one hand, there are the segmentation-based methods, where text images are segmented at word image level [8,28]; and, on the other hand, the segmentation-free methods, where words must be spotted from cropped text-lines, or full documents [2,26]. Moreover, according to the query modality, these methods can be classified either query-by-example (QbE) [25] or query-by-string (QbS) [3,13,28,33], being the second one, the more appealing from the user perspective.

The current trend in word spotting methods is based on learning a mapping function from the word images to a known word embedding spaces that can be

handcrafted [28,33] or learned by another network [8]. These family of approaches have demonstrated a very good performance in the QbS task. However, these methods focus on optimizing this mapping rather than making use of a ranking-based learning objective which is the final goal of a word spotting system.

In computer vision, siamese or triplet networks [31] trained with margin loss have been widely used for the image retrieval task. Despite their success in a large variety of tasks, instead of considering a real retrieval, these architectures act locally in pairs or triplets. To overcome this drawback, retrieval-based loss functions have emerged. These novel objective functions aim to optimize the obtained ranking. For instance, some works have been proposed to optimize the average precision (AP) metric [4,24]. Other works, such as Li *et al.* [18] suggested to only consider negative instances before positive ones to learn ranking-based models. Traditionally, word spotting systems are evaluated with the mean average precision (mAP) metric. However, this metric only takes into account a binary relevance function. Besides mAP, in these cases where a graded relevance score is required, information retrieval research has proposed the normalized Discounted Cumulative Gain (nDCG) metric. This is specially interesting for image retrieval and, in particular, word spotting because from the user perspective, the expected ranking list should follow such an order that is relevant to the user even though not being a perfect match. This metric has also been previously explored as a learning objective [30].

Taking into account the previous comments, the main motivation of this work is to analyze and evaluate two retrieval-based loss functions, namely Smooth-AP [4] and Smooth-nDCG, to train our QbS word spotting system based on the mAP and nDCG metrics respectively. We hypothesize that, with different purposes from the application perspective, training a word spotting system according to its retrieval list should lead to more robust embeddings. Figure 1 provides an overview of the expected behavior for each loss. On the one hand, the Smooth-AP loss expects to find the relevant images at the top ranking position *i.e.* these words with the same transcription as the query. On the other hand, Smooth-nDCG aims at obtaining a more interpretable list from the user’s perspective. For instance, the graded relevance function can consider the string edit distance or any other semantic similarity among strings depending on the final application.

To summarize, the main contributions of this work are:

- We propose a word spotting system which is trained solely with retrieval-based objective functions.
- We introduce a novel ranking loss function, namely, *Smooth-nDCG*, which is able to train our system according to a known relevance feedback. Therefore, we can present the results in a more pleasant way.
- We conduct extensive evaluation to demonstrate, on the one hand, the advantages and disadvantages of the proposed learning objectives and, on the other hand, the boost in word spotting performance for the QbS settings.

The rest of this paper is organized as follows. Section 2 reviews the previous works that are relevant to the current task. Afterwards, Section 3 presents the

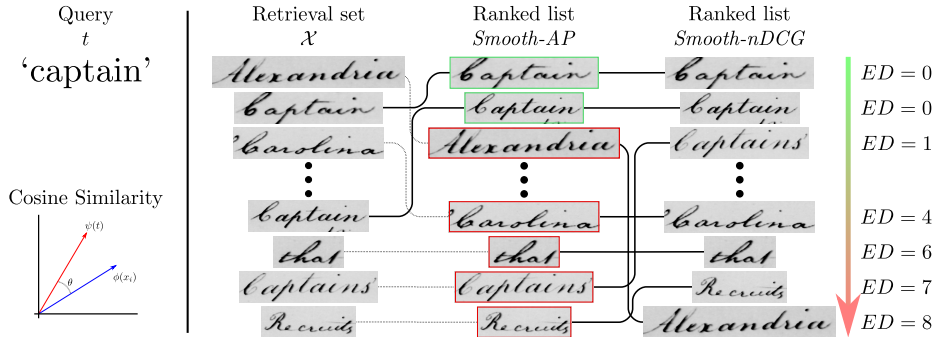


Fig. 1. Overview of the behavior of the proposed model. Given a query t and the retrieval set \mathcal{X} , the proposed word spotting system uses the cosine similarity to rank the retrieval set accordingly. Observe that Smooth-AP considers a binary relevance score whereas in the case of Smooth-nDCG, the ranked list is graded according to a non-binary relevance score such as the string edit distance.

embedding models for both, images and strings. Section 4 conducts an extensive evaluation on the proposed word spotting system. Finally, Section 5 draws the conclusions and the future work.

2 Related Work

2.1 Word Spotting

In this section, we introduce word spotting approaches that are relevant to the current work. Word spotting has been a hot research topic from its origin. The first successful attempts on using neural networks for word spotting were done by Frinken *et al.* [7]. They adapted the CTC Token passing algorithm to be applied to word spotting. In that work, the authors apply a BLSTM network to segmented text line images and then the CTC Token passing algorithm. Its main limitation is that it can only perform QbS tasks but not QbE.

Later, the research challenge evolved to integrate visual and textual information to perform the retrieval in a QbS setting. In this regard, one of the first works proposing this integration was Aldavert *et al.* [1]. In their work, the textual description is built based on n-gram histograms while the visual description is built by a pyramidal representation of bag of words [16]. Both representations are then projected to a common space by means of Latent Semantic Analysis (LSA) [6].

Another relevant work, introduced by Almazan *et al.* [3], propose a pyramidal histogram of characters, which they called PHOC, to represent in the same embedding space segmented word images and their corresponding text transcription. This shared representation enables exchange the input and output modalities in QbS and QbE scenarios. This seminal work inspired the current state-of-the-art methods [28,33,13] on using the PHOC representation. In [28],

Sudholt *et al.* propose the PHOCNet architecture to learn an image embedding representation of segmented words. This network architecture adapts the previous PHOC representation to be learned using a deep architecture. To deal with changing image sizes, the authors make use of a spatial pyramid pooling layer [9]. Similarly, Wilkinson *et al.* [33] train a triplet CNN followed a fully connected 2-layer network to learn an image embedding representation in the space of word embeddings. They evaluate two hand-crafted embeddings, the PHOC and the discrete cosine transform. To learn a shared embedding space, they used the cosine loss to train the network.

Krishnan *et al.* [13] also proposed a deep neuronal architecture to learn a common embedding space for visual and textual information. The authors use a pre-trained CNN on synthetic handwritten data for segmented word images [15] and the PHOC as attribute-based text representation. Later, they presented an improved version of this work in [14]. On the one hand, the architecture is improved to the ResNet-34. they also exploit the use of synthetic word images at test time.

However, all these methods focus on training architectures to rank first relevant images but without ranking them into a retrieval list.

More recently, Gomez *et al.* [8] faced the problem of obtaining a more appealing retrieval list. First, they proposed a siamese network architecture that is able to learn a string embedding that correlates with the Levenshtein edit distance [17]. Second, they train a CNN model inspired from [11] to train an image embedding close to the corresponding string embedding. Thus, even though they share our motivation, they do not exploit this ranking during training.

2.2 Ranking metrics

In word spotting and, more generally in information retrieval, several metrics have been carefully designed to evaluate the obtained rankings. We briefly define the two main ones, that are considered in this work.

Mean Average Precision: Word spotting performance has been traditionally measured by the *mean Average Precision* (mAP), a classic information retrieval metric [27]. The *Average Precision* given a query q (AP_q) is formally defined as the mean

$$AP_q = \frac{1}{|\mathcal{P}_q|} \sum_{n=1}^{|\Omega_q|} P@n \times r(n), \quad (1)$$

where $P@n$ is the precision at n , $r(n)$ is a binary function on the relevance of the n -th item in the returned ranked list, \mathcal{P}_q is the set of all relevant objects with regard the query q and Ω_q is the set of retrieved elements from the dataset. Then, the mAP is defined as:

$$\text{mAP} = \frac{1}{Q} \sum_{q=1}^Q AP_q, \quad (2)$$

where Q is the number of queries.

Normalized Discounted Cumulative Gain: In information retrieval, the *normalized Discounted Cumulative Gain* (nDCG) is used to measure the performance on such scenarios where instead of a binary relevance function, we have a graded relevance scale. The main idea is that highly relevant elements appearing lower in the retrieval list should be penalized. The *Discounted Cumulative Gain* (DCG) for a query q is defined as

$$\text{DCG}_q = \sum_{n=1}^{|\Omega_q|} \frac{r(n)}{\log_2(n+1)}, \quad (3)$$

where $r(n)$ is a graded function on the relevance of the n -th item in the returned ranked list and Ω_q is the set of retrieved elements as defined above. However, to allow a fair comparison among different queries, a normalized version was proposed and defined as

$$\text{nDCG}_q = \frac{\text{DCG}_q}{\text{IDCG}_q}, \quad (4)$$

where IDCG_q is the ideal discounted cumulative gain, *i.e.* assuming a perfect ranking according to the relevance function. It is formally defined as

$$\text{IDCG}_q = \sum_{n=1}^{|\Lambda_q|} \frac{r(n)}{\log_2(n+1)} \quad (5)$$

where Λ_q is the ordered set according the relevance function.

3 Architecture

3.1 Problem formulation

Let $\{\mathcal{X}, \mathcal{Y}\}$ be a word image dataset, containing word images $\mathcal{X} = \{x_i\}_{i=0}^N$, and their corresponding transcription strings \mathcal{Y} . Let \mathcal{A} be the alphabet containing the allowed characters. Given the word images \mathcal{X} as a retrieval set or gallery for searching purposes on the one hand, and a given text string t such that its characters $t_i \in \mathcal{A}$ on the other hand; the proposed embedding functions $\phi(\cdot)$ and $\psi(\cdot)$ for word images and text strings respectively, have the objective to map its input in such a space that a given similarity function $S(\cdot, \cdot)$ is able to retrieve a ranked list with the relevant elements. Traditionally, the evaluation of word spotting divides this list in two partitions, namely the positive or relevant word images *i.e.* their transcription matches with the query t , and the negative or non-relevant word images. Thus, the aim of a word spotting system is to rank the positive elements at the beginning of the retrieval list. However, from the user perspective, the non-relevant elements might be informative enough to require them to follow some particular order. Therefore, we formulate the word spotting

problem in terms of a relevance score $R(\cdot, \cdot)$ for each element in the list. Finally, given a query t we can formally define this objective as

$$S(\phi(x_i), \psi(t)) > S(\phi(x_j), \psi(t)) \iff R(y_i, t) > R(y_j, t), \quad (6)$$

where $x_i, x_j \in \mathcal{X}$ are two elements in the retrieved list and $y_i, y_j \in \mathcal{Y}$ their corresponding transcriptions.

3.2 Encoder Networks

As already explained, the proposed word spotting system consists of a word image encoding $\phi(\cdot)$ and a textual encoding $\psi(\cdot)$. The outputs of these encoding functions are expected to be in the same space.

Word image encoding: Given a word image $x_i \in \mathcal{X}$, it is firstly resized into a fixed height, but we keep the original aspect ratio. As the backbone of our image encoding, we use a ImageNet pre-trained ResNet-34 [10] network. As a result of the average pooling layer, our model is able to process images of different widths. Finally, we L_2 -normalize the final 64D embedding.

String encoder: The string encoder embeds a given query string t in a 64D space. Firstly, we define a character-wise embedding function $e: \mathcal{A} \rightarrow \mathbb{R}^m$. Thus, the first step of our encoder is to embed each character $c \in t$ into the character latent space. Afterwards, each character is feed into a Bidirectional Gated Recurrent Unit (GRU) [5] with two layers. In addition, a linear layer takes the last hidden state of each direction to generate our final word embedding. Finally, we also L_2 -normalize the final 64D embedding.

Although both embeddings can be compared by means of any arbitrary similarity measure $S(\cdot, \cdot)$, in this work we decided to use the cosine similarity between two embeddings v_q and v_i , which is defined as:

$$S(v_q, v_i) = \frac{v_q \cdot v_i}{\|v_q\| \|v_i\|}. \quad (7)$$

From now on and for the sake of simplicity, given a query q its corresponding similarity against the i -th element of the retrieval set is denoted as $s_i = S(v_q, v_i)$.

3.3 Learning Objectives

As already stated in the introduction, we analyze two learning objectives which provide supervision at the retrieval list level rather than at sample, pair or triplet level. The proposed losses are inspired by the classical retrieval evaluation metrics introduced in Section 2.

Let us first define some notations and concepts.

Ranking Function (\mathcal{R}). Following the notation introduced in [22], these information retrieval metrics can be reformulated by means of the following ranking function,

$$\mathcal{R}(i, \mathcal{C}) = 1 + \sum_{j=1}^{|\mathcal{C}|} \mathbb{1}\{D_{ij} < 0\}, \quad (8)$$

where \mathcal{C} is any set (such as Ω_q or \mathcal{P}_q), $D_{ij} = s_i - s_j$ and $\mathbb{1}\{\cdot\}$ is an Indicator function.

However, with this formulation, we are not able to optimize following the gradient based optimization methods and an smooth version of the Indicator is required. Even though several approximations exist, in this work we followed the one proposed by Quin *et al.* [22], which make use of the sigmoid function

$$\mathcal{G}(x; \tau) = \frac{1}{1 + e^{-\frac{x}{\tau}}}. \quad (9)$$

Smooth-AP: The smoothed approximation of AP, namely Smooth-AP and proposed by Brown *et al.* [4], has shown a huge success on image retrieval. There, the authors replace the $P@n \times r(n)$ term in Equation 1 by the ranking function and the exact AP equation becomes

$$AP_q = \frac{1}{|\mathcal{P}_q|} \sum_{i \in \mathcal{P}_q} \frac{1 + \sum_{j \in \mathcal{P}_q, j \neq i} \mathbb{1}\{D_{ij} < 0\}}{1 + \sum_{j \in \Omega_q, j \neq i} \mathbb{1}\{D_{ij} < 0\}}. \quad (10)$$

Therefore, with this notation we can directly obtain an smooth approximation making use of Equation 9 as

$$AP_q \approx \frac{1}{|\mathcal{P}_q|} \sum_{i \in \mathcal{P}_q} \frac{1 + \sum_{j \in \mathcal{P}_q, j \neq i} \mathcal{G}(D_{ij}; \tau)}{1 + \sum_{j \in \Omega_q, j \neq i} \mathcal{G}(D_{ij}; \tau)}. \quad (11)$$

Averaging this approximation for all the queries in the batch, we can define our loss as

$$\mathcal{L}_{AP} = 1 - \frac{1}{Q} \sum_{i=1}^Q AP_q, \quad (12)$$

where Q is the number of queries.

Smooth-nDCG: Following the same idea as above, we replace the n -th position in Equation 3 by the ranking function, since it defines the position that the i -th element of the retrieved set, and the DCG metric is expressed as

$$DCG_q = \sum_{i \in \Omega_q} \frac{r(i)}{\log_2 \left(2 + \sum_{j \in \Omega_q, j \neq i} \mathbb{1}\{D_{ij} < 0\} \right)}, \quad (13)$$

where r is the same graded function used in Equation 3 but evaluated at element i . Therefore, the corresponding smooth approximation is

$$DCG_q \approx \sum_{i \in \Omega_q} \frac{r(i)}{\log_2 \left(2 + \sum_{j \in \Omega_q, j \neq i} \mathcal{G}(D_{ij}; \tau) \right)} \quad (14)$$

when replacing the indicator function by the sigmoid one.

The smooth-nDCG is then defined by replacing the original DCG_q by its smooth approximation in Equation 4 and the loss \mathcal{L}_{nDCG} is defined as

$$\mathcal{L}_{nDCG} = 1 - \frac{1}{Q} \sum_{i=1}^Q nDCG_q, \quad (15)$$

Algorithm 1 Training algorithm for the proposed model.

Input: Input data $\{\mathcal{X}, \mathcal{Y}\}$; alphabet \mathcal{A} ; max training iterations T

Output: Networks parameters $\Theta = \{\Theta_\phi, \Theta_\psi\}$.

1: **repeat**

2: Get word images $X = \{x_i\}_{i=1}^{N_B}$ and its corresponding transcription $Y = \{y_i\}_{i=1}^{N_B}$

3: $\mathcal{L} \leftarrow \mathcal{L}_{img} + \mathcal{L}_{str} + \mathcal{L}_{cross} + \alpha \frac{1}{N_B} \sum_{i=1}^{N_B} \mathcal{L}_{L_1}$

4: $\Theta \leftarrow \Theta - \Gamma(\nabla_{\Theta} \mathcal{L})$

5: **until** Max training iterations T

where Q is the number of queries.

3.4 End-to-End training

Considering a batch of size N_B of word images $X = \{x_i\}_{i=1}^{N_B}$ and their corresponding transcriptions $Y = \{y_i\}_{i=1}^{N_B}$, the proposed word image and string encoder is trained considering all the elements in both, the query and the retrieval set. Bearing in mind that Smooth-AP cannot be properly used to train the string encoder alone, we combine the Smooth-AP and Smooth-nDCG loss functions into the following three loss functions:

$$\mathcal{L}_{img} = \mathcal{L}_{AP}(X) + \mathcal{L}_{nDCG}(X) \quad (16)$$

$$\mathcal{L}_{str} = \mathcal{L}_{nDCG}(Y) \quad (17)$$

$$\mathcal{L}_{cross} = \mathcal{L}_{AP}(Y, X) + \mathcal{L}_{nDCG}(Y, X) \quad (18)$$

Moreover, the L_1 -loss \mathcal{L}_{L_1} between each image embedding and its corresponding word embedding is used to force both models to lay in the same embedding space. This loss is multiplied by a parameter α that we set experimentally to 0.5. Note that the gradients of this loss will only update the weights of the word image encoder. Algorithm 1 depicts the explained training algorithm in the situation in which both losses are considered. $\Gamma(\cdot)$ denotes the optimizer function.

4 Experimental Evaluation

In this section, we present an exhaustive evaluation of the proposed model in the problem of word spotting. All the code necessary to reproduce the experiments is available at github.com/priba/ndcg_wordspotting_pytorch using the PyTorch framework.

4.1 Implementation details

The proposed model is trained for 50 epochs where in each epoch 15,000 samples are drawn by means of a random weighted sampler. This setting ensures that the

data is balanced among the different classes during the training of the model. In addition, data augmentation is applied in the form of a random affine transformation. The possible transformations have been kept small to ensure that the text is not altered. Thus, we only allow rotations and shear up to 5 degrees and a scale factor in the range of 0.9 and 1.1.

For all the experiments the Adam optimizer [12] has been employed. The learning rate, starting from 1e-4 has been decreased by a factor of 0.25 at epochs 25 and 40.

For evaluation purposes, we used the classic ranking metrics introduced in Section 2, *i.e.* mAP and nDCG considering the full retrieval set. In particular, in our setup we define the following relevance function $r(n)$ to the nDCG metric, these words with edit distances 0, 1, 2, 3 and 4 receive a score of 20, 15, 10, 5 and 3. In addition, the smooth-nDCG loss is trained with the following relevance function,

$$r(n; \gamma) = \max(0, \gamma - \text{Lev}(q, y_n)), \quad (19)$$

where q and $y_n \in \mathcal{Y}$ are the transcriptions of the query at the n -th ranking of the retrieval list and $\text{Lev}(\cdot, \cdot)$ corresponds to the Levenshtein distance [17] between two strings. Moreover, γ is a hyperparameter that has been set experimentally to 4 in our case of study.

4.2 Experimental discussion

The proposed model has been evaluated in two different datasets. First, the GW dataset, which is composed of handwriting word images and, second, the IIIT5K dataset, which is composed of words cropped from real scenes.

George Washington (GW) [23]. This database is based on handwritten letters written in English by George Washington and his associates during the American Revolutionary War in 1755¹. It consists of 20 pages with a total of 4,894 handwritten words. Even though several writers were involved, it presents small variations in style and only minor signs of degradation. There is not official partition for the GW dataset, therefore, we follow the evaluation adopted by Almazan *et al.* [3]. Thus, the dataset is splitted in two sets at word level containing 70% of the words for training purpose and the remaining 25% for test. This setting is repeated four times following a cross validation setting. Finally, we report the average result of these four runs.

The IIIT 5K-word dataset (IIIT5K) dataset [21]. This dataset provides 5,000 cropped word images from scene texts and born digital images obtained from Google Image engine search. The authors of this dataset provide an official partition containing two subsets of 2,000 and 3,000 images for training and testing purposes. In addition, the each word is associated with two lexicon subsets of 50 and 1,000 words, respectively. Moreover, a global lexicon [32] of more than

¹ George Washington Papers at the Library of Congress from 1741-1799, Series 2, Letterbook 1, pages 270-279 and 300-309, <https://www.loc.gov/collections/george-washington-papers/about-this-collection/>

half a million words can be used for word recognition. In this work, none of the provided lexicons have been used.

For experimental purposes, we have evaluated three different combinations of the introduced learning objectives, namely, Smooth-AP, Smooth-nDCG and Join, *i.e.* a combination of the previous losses.

Table 1. Mean of ranking metrics, mAP and nDCG, for state-of-the-art query-by-string (QbS) and query-by-example (QbE) methods in the GW and IIIT5K datasets.

Method	GW				IIIT5K			
	QbS		QbE		QbS		QbE	
	mAP	nDCG	mAP	nDCG	mAP	nDCG	mAP	nDCG
Aldavert <i>et al.</i> [1]	56.54	-	-	-	-	-	-	-
Frinken <i>et al.</i> [7]	84.00	-	-	-	-	-	-	-
Almazán <i>et al.</i> [3]	91.29	-	93.04	-	66.24	-	63.42	-
Goméz <i>et al.</i> [8]	91.31	-	-	-	-	-	-	-
Sudholt <i>et al.</i> [28]	92.64	-	96.71	-	-	-	-	-
Krishnan <i>et al.</i> [13]	92.84	-	94.41	-	-	-	-	-
Wilkinson <i>et al.</i> [33]	93.69	-	97.98	-	-	-	-	-
Sudholt <i>et al.</i> [29]	98.02	-	97.78	-	-	-	-	-
Krishnan <i>et al.</i> [14]	98.86	-	98.01	-	-	-	-	-
Smooth-AP <i>et al.</i> [4]	96.79	88.99	97.17	87.64	81.25	80.86	80.60	76.13
Smooth-nDCG	98.25	96.41	97.94	94.27	88.99	90.63	87.53	85.11
Join	98.38	96.40	98.09	94.27	89.14	90.64	87.60	85.16

Table 1 provides a comparison with the state-of-the-art techniques for word spotting. Most of these techniques have been only evaluated in the handwritten case corresponding to the GW dataset. The best performing model for QbS in the GW dataset is the method proposed by Krishnan *et al.* [14] that obtains a mAP slightly better than our Join setting. Even though, our proposed model has been pre-trained with ImageNet, Krishnan *et al.* makes use of a huge dataset of 9M synthetic word images. Besides the good performance for QbS task, our model is able to perform slightly better in the QbE task. From the same table, we can observe that our model is able to generalize to real scene word images. In such dataset, we outperform the work of Almazán *et al.* by more than 20 points for both tasks, QbS and QbE.

Observe that exploiting the Smooth-nDCG loss, our model is able to enhance the performance of the architecture trained with the Smooth-AP objective function alone. This remarks the importance of not only considering those images whose transcription matches with the query word. In addition, in terms of the nDCG, the performance is also boosted by this loss. Thus, the results are more appealing to the final user following the pre-defined relevance function, in this example, the string edit distance.

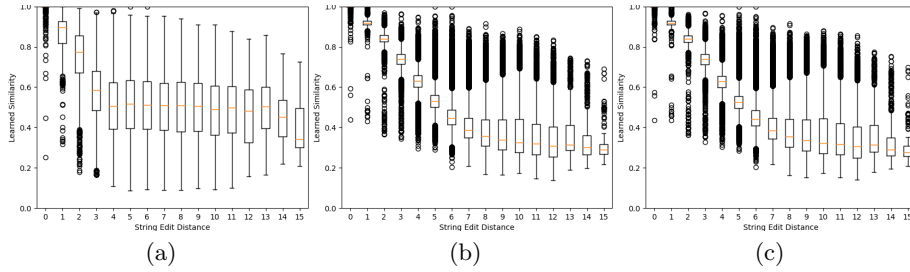


Fig. 2. Box plots for (a) smooth-AP; (b) smooth-nDCG; (c) Join, losses.

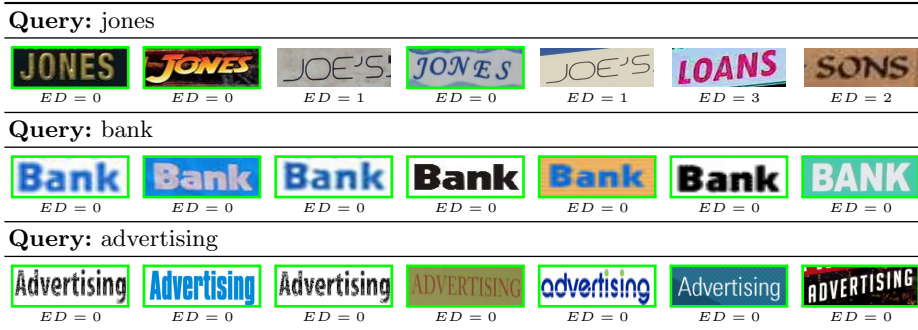


Fig. 3. Qualitative retrieval results for the IIIT5K dataset using the model trained with the Join loss. In green, the exact matches.

The achieved performance can be explained by the box plots depicted in Figure 2. This figure shows the correlation between the real string edit distance and the learned similarity of two words. On the one hand, note that the model trained by the smooth-AP loss is able to specialize on detecting which is the matching image given a query. On the other hand, both nDCG or Join losses are able to describe a better ranking when increasing the real string edit distance.

Figures 3 and 4 demonstrates qualitatively the performance of the proposed setting. Observe that the proposed model is able to rank in the first positions the exact match given the query word. Moreover, introducing the Smooth-nDCG we observe that the edit distance is in general smaller than in the Smooth-AP case.

Finally, Figure 5 provides an overview of the average edit distance of the top- n results given a query. For instance, the Ideal case, tells us that considering the ground-truth, in average, all the retrieved images in the top-50 have an edit distance smaller than 3.5. Here, we can clearly identify that both learning objectives Smooth-nDCG and Join are able to close the gap between the Ideal case and the Smooth-AP loss.



Fig. 4. Qualitative results for the GW dataset. In green, the exact matches.

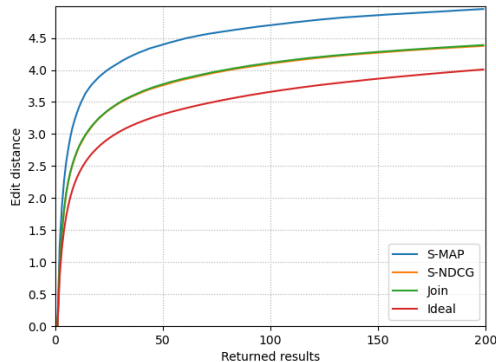


Fig. 5. Average edit distance among the top- n returned results. Note that Smooth-nDCG and Join overlaps in the plot being Smooth-nDCG slightly closer to the Ideal case.

5 Conclusions

In this work, we have presented a word spotting framework completely based on ranking-based losses. The proposed approach learns directly from the retrieval list rather than pairs or triplets as most of the state-of-the-art methodologies. In addition, we do not require any prelearned word embedding. From the application point of view, we have shown the competitive performance of our model against the state-of-the-art methods for word spotting in handwritten and real scene text images. Overall, we have demonstrated the importance of considering not only the corresponding image/transcription pair but also, they relation between the different elements in the batch thanks to a graded relevance score.

As future work, we plan to perform an exhaustive evaluation on the different hyperparameters of the proposed smooth-nDCG objective such as, the relevance and the indicator functions. Moreover, the final loss can be weighted between the smooth-AP and the smooth-nDCG losses. Finally, we would like to explore how this framework extends to other multi-modal retrieval tasks.

Acknowledgment

This work has been partially supported by the Spanish projects RTI2018-095645-B-C21, and FCT-19-15244, and the Catalan projects 2017-SGR-1783, the Culture Department of the Generalitat de Catalunya, and the CERCA Program / Generalitat de Catalunya.

References

1. Aldavert, D., Rusiñol, M., Toledo, R., Lladós, J.: Integrating visual and textual cues for query-by-string word spotting. In: Proceedings of the International Conference on Document Analysis and Recognition. pp. 511–515 (2013)
2. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Segmentation-free word spotting with exemplar svms. *Pattern Recognition* **47**(12), 3967–3978 (2014)
3. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(12), 2552–2566 (2014)
4. Brown, A., Xie, W., Kalogeiton, V., Zisserman, A.: Smooth-AP: Smoothing the path towards large-scale image retrieval. In: Proceedings of the European Conference on Computer Vision (2020)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of the NeurIPS Workshop on Deep Learning (2014)
6. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science* **41** pp. 391–407 (1990)
7. Frinken, V., Fischer, A., Manmatha, R., Bunke, H.: A novel word spotting method based on recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34**(2), 211–224 (2011)
8. Gómez, L., Rusiñol, M., Karatzas, D.: LSDE: Levenshtein space deep embedding for query-by-string word spotting. In: Proceedings of the International Conference on Document Analysis and Recognition. vol. 1, pp. 499–504 (2017)
9. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37**(9), 1904–1916 (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
11. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint **1406.2227** (2014)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Krishnan, P., Dutta, K., Jawahar, C.: Deep feature embedding for accurate recognition and retrieval of handwritten text. In: Proceedings of the International Conference on Frontiers in Handwriting Recognition. pp. 289–294 (2016)
14. Krishnan, P., Dutta, K., Jawahar, C.: Word spotting and recognition using deep embedding. In: Proceedings of the International Workshop on Document Analysis Systems. pp. 1–6 (2018)
15. Krishnan, P., Jawahar, C.V.: Matching handwritten document images. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Proceedings of the European Conference on Computer Vision. pp. 766–782 (2016)
16. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 2169–2178 (2006)

17. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* **10**(8), 707–710 (1966)
18. Li, Z., Min, W., Song, J., Zhu, Y., Jiang, S.: Rethinking ranking-based loss functions: Only penalizing negative instances before positive ones is enough. *arXiv preprint* (2021)
19. Manmatha, R., Chengfeng Han, Riseman, E.M.: Word spotting: a new approach to indexing handwriting. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 631–637 (1996)
20. Manmatha, R., Han, C., Riseman, E.M., Croft, W.B.: Indexing handwriting using word matching. In: *Proceedings of the ACM International Conference on Digital Libraries*. pp. 151–159 (1996)
21. Mishra, A., Alahari, K., Jawahar, C.V.: Scene text recognition using higher order language priors. In: *Proceedings of the British Machine Vision Conference* (2012)
22. Qin, T., Liu, T.Y., Li, H.: A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* **13**(4), 375–397 (2010)
23. Rath, T.M., Manmatha, R.: Word spotting for historical documents. *International Journal on Document Analysis and Recognition* **9**(2-4), 139–152 (2007)
24. Revaud, J., Almazán, J., Rezende, R.S., Souza, C.R.d.: Learning with average precision: Training image retrieval with a listwise loss. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 5107–5116 (2019)
25. Rusiñol, M., Aldavert, D., Toledo, R., Lladós, J.: Browsing heterogeneous document collections by a segmentation-free word spotting method. In: *Proceedings of the International Conference on Document Analysis and Recognition*. pp. 63–67 (2011)
26. Rusiñol, M., Aldavert, D., Toledo, R., Lladós, J.: Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition* **48**(2), 545–555 (2015)
27. Rusiñol, M., Lladós, J.: A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *International Journal on Document Analysis and Recognition* **12**(2), 83–96 (2009)
28. Sudholt, S., Fink, G.A.: PHOCNet: A deep convolutional neural network for word spotting in handwritten documents. In: *Proceedings of the International Conference on Frontiers in Handwriting Recognition*. pp. 277–282 (2016)
29. Sudholt, S., Fink, G.A.: Evaluating word string embeddings and loss functions for cnn-based word spotting. In: *Proceedings of the International Conference on Document Analysis and Recognition*. vol. 1, pp. 493–498 (2017)
30. Valizadegan, H., Jin, R., Zhang, R., Mao, J.: Learning to rank by optimizing ndcg measure. In: *Advances in Neural Information Processing Systems*. vol. 22, pp. 1883–1891 (2009)
31. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* **10**, 207–244 (2009)
32. Weinman, J.J., Learned-Miller, E., Hanson, A.: Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(10), 1733–1746 (2009)
33. Wilkinson, T., Brun, A.: Semantic and verbatim word spotting using deep neural networks. In: *Proceedings of the International Conference on Frontiers in Handwriting Recognition*. pp. 307–312 (2016)