Learning Term Discrimination

Jibril Frej, Philippe Mulhem, Didier Schwab, Jean-Pierre Chevallet

Learning Term Discrimination

Jibril Frej
jibril.frej@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
* Institute of Engineering Univ. Grenoble Alpes

Philippe Mulhem
philippe.mulhem@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
* Institute of Engineering Univ. Grenoble Alpes

Didier Schwab
didier.schwab@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
* Institute of Engineering Univ. Grenoble Alpes

Jean-Pierre Chevallet
jean-pierre.chevallet@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG
* Institute of Engineering Univ. Grenoble Alpes

ABSTRACT

Document indexing is a key component for efficient information retrieval (IR). After preprocessing steps such as stemming and stop-word removal, document indexes usually store term-frequencies (tf). Along with tf (that only reflects the importance of a term in a document), traditional IR models use term discrimination values (TDVs) such as inverse document frequency (idf) to favor discriminative terms during retrieval. In this work, we propose to learn TDVs for document indexing with shallow neural networks that approximate traditional IR ranking functions such as TF-IDF and BM25. Our proposal outperforms, both in terms of nDCG and recall, traditional approaches, even with few positively labelled query-document pairs as learning data. Our learned TDVs, when used to filter out terms of the vocabulary that have zero discrimination value, allow to both significantly lower the memory footprint of the inverted index and speed up the retrieval process (BM25 is up to 3 times faster), without degrading retrieval quality.

KEYWORDS

Information Retrieval, Shallow Neural Networks, Document Indexing, Term Discrimination Value

1 INTRODUCTION

Document indexing for information retrieval (IR) usually consists in associating each document of a collection with a set of weighted terms reflecting its information content. To this end, a term discrimination value (TDV) is used to represent the usefulness of a term as a discriminator among documents [13]. However, traditional IR systems make little use of TDVs during indexing. The only exception is stopword removal which considers that stop-words have null discrimination value and removes them from document representations. Stop-word removal also speeds up the retrieval process when using an inverted index since it removes stop-words that have long posting lists.

Related Work. Several methods have been proposed to compute TDVs, such as using the density of the document vector space [14] or the covering coefficient of documents [2]. Nowadays, the most common approaches in traditional IR models for computing TDVs are to use either the inverted document frequency (idf) [11] or a smoothing method such as Bayesian smoothing using Dirichlet prior [16]. Recently, Roy et al. [12] proposed to select discriminative terms to enhance query expansion methods based on pseudo-relevance feedback. However, these approaches use TDVs only at retrieval time and not during indexation. Inspired by stop-word removal, we suggest that using supervised learning to remove non discriminative terms at indexation can speed up the retrieval process with no deterioration of retrieval quality.

Our Contributions. In this work, we propose to learn TDVs in a supervised setting using a shallow neural network and word embeddings. In order to have TDVs adapted to traditional IR ranking functions, we propose to learn TDVs by optimizing the ranking of traditional IR models. However, components of these models such as term frequency (tf) or inverse document frequency (idf) are not differentiable in the setting in which neural networks are commonly used (sequences of word embeddings processed by CNN, RNN or Transformer Layers). This non-differentiability makes impossible the use of gradient descent-based optimization methods required by neural networks. Hence, we propose a setting that uses bag of words (BoWs) as sparse vectors to have differentiable tf and ℓ_1-norm as an approximation to the ℓ_0-norm to have a differentiable approximation of the idf. Hence, we learn TDVs to optimize differentiable approximations of traditional IR ranking functions. Since we are using a shallow neural network with few parameters, our models do not need large amounts of positively labelled query-documents pairs to outperform traditional IR models. Additionally, we remove posting lists associated with terms having zero TDV from the inverted index in order to significantly enhance retrieval speed.

In short, our contributions are :

- A new framework for differentiable traditional IR;
- Differentiable versions of IR functions to learn TDVs;
- A significant speed up retrieval obtained by removing posting lists associated to terms with zero TDV;

2 LEARNING TERM DISCRIMINATION

To learn TDVs adapted to traditional IR ranking functions using neural networks, we propose the following strategy:

(1) Make traditional IR ranking functions compatible with neural networks by using matrix operations that are differentiable with respect to the inverted index (Section 2.1);

, 2020,

(2) Introduce a shallow neural network to compute TDVs and a method to include TDVs into the inverted index (Section 2.2);
(3) Use the differentiable functions proposed in Section 2.1 to learn TDVs adapted to traditional IR ranking functions using a supervised shallow neural network (Section 2.3);

2.1 Differentiable traditional IR

All operations used by traditional IR ranking function can be derived from the inverted index. Given a vocabulary V and a collection C, the inverted index can be represented as a sparse matrix $S \in \mathbb{R}^{|V|\times|C|}$. Each element of S corresponds to the term frequency (tf) of a term $t \in V$ with respect to (w.r.t) a document $d \in C$: $S_{t,d} = \text{tf}_{td}$. Columns of S (denoted as $S_{:,d}$) correspond to the BoW representations of documents in C and rows of S (denoted as $S_{t,:}$) correspond to the posting lists of terms in V. Let $Q \in \mathbb{N}^{|V|}$ denote the BoW representation of a query q.

2.1.1 TF-IDF. Using matrix operations over S, the TF-IDF ranking function between a query q and a document d can be formulated as:

$$\text{TF-IDF}(q,d) = \sum_{t\in q} \text{tf}_{td}\ \text{idf}_t = Q^\intercal \cdot \left(S_{:,d} \odot \text{IDF}\right), \tag{1}$$

where $\odot$ denotes the element wise (or Hadamard) product and $\text{IDF} \in \mathbb{R}^{|V|}$ denotes the vector containing inverse document frequencies (idf) of all terms V. idf can be derived from S using the ℓ_0-norm to compute document frequencies (df):

$$\text{idf}_t = \log \frac{|C|+1}{\text{df}_t} = \log \frac{|C|+1}{\ell_0(S_{t,:})}. \tag{2}$$

To be able to have TDVs adapted to traditional IR ranking functions, we want such functions to be differentiable w.r.t elements of S. However, the ℓ_0-norm is non differentiable. Consequently, we propose to redefine idf using ℓ_1 which is a good approximation to ℓ_0 [10]. If we replace ℓ_0 with ℓ_1, the obtained idf will be negative for terms such that $\ell_0(S_{t,:}) > |C|+1$ which would violate the Term Frequency Constraint, a desirable property of retrieval formula [4]. To ensure positives idfs, we propose a maximum normalization:

$$\widetilde{\text{idf}_t} = \log \frac{\max_{\{t'\in V\}} \ell_1(S_{t',:}) + 1}{\ell_1(S_{t,:})}. \tag{3}$$

Using $\widetilde{\text{idf}_t}$, we have the following differentiable approximation of the TF-IDF formula, denoted as $\widetilde{\text{TF-IDF}}$:

$$\widetilde{\text{TF-IDF}}(q,d) = Q^\intercal \cdot \left(S_{:,d} \odot \widetilde{\text{IDF}}\right) = \sum_{t\in q} S_{t,d}\ \widetilde{\text{idf}_t}. \tag{4}$$

where $\widetilde{\text{IDF}} \in \mathbb{R}^{|V|}$ is the vector containing $\widetilde{\text{idf}_t}$ of all terms in V.

2.1.2 BM25. We can also define a differentiable approximation of the BM25 ranking formula using $\widetilde{\text{IDF}}$:

$$\widetilde{\text{BM25}}(q,d) = Q^\intercal \cdot \widetilde{\text{IDF}} \odot \left(S_{:,d}\,(k_1+1)\right) \\ ./\left(S_{:,d} + k_1\left(1 - b + b\frac{|d|}{avgdl}\right)\mathbb{1}_{|V|}\right), \tag{5}$$

where k_1 and b are parameters of BM25 and *avgdl* denotes the average length of documents in C. $./$ is an element wise (or Hadamard) division and $\mathbb{1}_{|V|}$ is a vector of dimension $|V|$ with all elements equal to one. Both $|d|$ and *avgdl* are differentiable w.r.t the elements of S: $|d| = \ell_1(S_{:,d})$ and $avgdl = \sum_{d\in C} \ell_1(S_{:,d})/|C|$.

2.1.3 Dirichlet Language Model. We also propose a differentiable language model with Dirichlet prior smoothing [16]:

$$\begin{aligned}\text{LM}(q,d) &= \sum_{t\in q} \log\left(1 + \frac{\text{tf}_{td}}{\mu p(t|C)}\right) + |q|\log\alpha_d \\ &= Q^\intercal \cdot \left[\log\left(\mathbb{1}_{|V|} + \left(S_{:,d}./(\mu P_C)\right)\right)\right. \\ &\quad \left. + |q|\log(\alpha_d)\mathbb{1}_{|V|}\right],\end{aligned} \tag{6}$$

where μ is a parameter of LM, $\alpha_d = \frac{\mu}{|d|+\mu}$ is a document dependent constant and $P_C \in \mathbb{R}^{|V|}$ is the vector containing the probability of a term given the collection language models for all terms in the vocabulary: $\forall t \in V, P_{C_t} = p(t|C) = \sum_{d\in C} S_{t,d} / \sum_{t'\in V}\sum_{d\in C} S_{t',d}$.

2.2 Shallow neural network for learning TDVs

To have a model that requires few training data, we propose to compute the TDVs using a shallow neural network composed of a single linear layer and the Rectified Linear Unit (ReLU) non linearity: $\text{tdv}_t = \text{ReLU}(w_t^\intercal \cdot w + b) = \max(0, w_t^\intercal \cdot w + b)$ where w_t is the word embedding of term t, and w and b are parameters of the neural network. We employ ReLU activation function to ensure that the TDV is positive (as negative TDVs can violate the Term Frequency Constrain) and to be able to have terms with zero TDV that we can remove from the inverted index. We redefine the inverted index S using TDVs the following way:

$$S'_{t,d} = \text{tf}_{td}\ \text{tdv}_t = \text{tf}_{td}\ \text{ReLU}(w_t^\intercal \cdot w + b). \tag{7}$$

With this definition, we ensure that if a term t has zero discrimination value ($\text{tdv}_t = 0$), the row in S associated to t is filled with zeros, therefore it's posting list is empty and can be removed from the inverted index.

2.3 Learning TDVs with differentiable IR

To learn TDVs that optimize the score of traditional IR ranking formulae, we simply replace S in Equations (4), (5) and (6) by S':

$$\text{TDV-TF-IDF}(q,d) = Q^\intercal \cdot \left(S'_{:,d} \odot \widetilde{\text{IDF}'}\right); \tag{8}$$

$$\begin{aligned}\text{TDV-BM25}(q,d) &= Q^\intercal \cdot \widetilde{\text{IDF}'} \odot \left(S'_{:,d}\,(k_1+1)\right) \\ &\quad ./\left(S_{:,d} + k_1\left(1 - b + b\frac{|d|'}{avgdl'}\right)\mathbb{1}_{|V|}\right);\end{aligned} \tag{9}$$

$$\begin{aligned}\text{TDV-LM}(q,d) &= Q^\intercal \cdot \left[\log(\mathbb{1}_{|V|} + (S'_{:,d}./(\mu P'_C))\right. \\ &\quad \left. + |q|\log(\alpha'_d)\mathbb{1}_{|V|}\right];\end{aligned} \tag{10}$$

where $\widetilde{\text{IDF}'}$, $|d|'$, $avgdl'$ and α'_d denote respectively $\widetilde{\text{IDF}}$, $|d|$, $avgdl$ and α_d computed with S' instead of S. Scores computed by these ranking functions are differentiable w.r.t parameters w and b (see

Figure 1). Consequently, we can use gradient descent-based optimization methods to update w and b in order to compute TDVs adapted to traditional IR ranking functions.

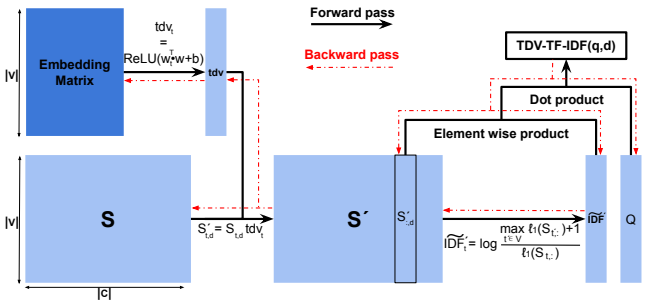

Figure 1: Architecture of TDV-TF-IDF. All operations are differentiable and gradients can be back-propagated from the final score to w and b.

2.4 Training

We use the pairwise hinge loss function as our ranking objective:

$$\mathcal{L}_{\text{Hinge}}(f, q, d^+, d^-) = \max(0, 1 - f(q, d^+) + f(q, d^-)), \quad (11)$$

where f is a differentiable ranking function for IR and d^+ is a document more relevant than d^- w.r.t query q. To ensure that the ranking functions produce terms with zero discrimination values, we also use a sparsity objective during training. To do so, we minimize the ℓ_1-norm of the document BOWs representation as suggested by Zamani et al. [15]. The final loss function is defined as follows:

$$(1-\lambda)\mathcal{L}_{\text{Hinge}}(f, q, d^+, d^-) + \lambda(\ell_1(Sf'_{:,d^+}) + \ell_1(Sf'_{:,d^-})), \quad (12)$$

where $\lambda \in [0, 1]$ is the regularization hyper-parameter and Sf' is the inverted index matrix computed by f.

3 EXPERIMENTS

3.1 Collections

As mentioned previously, our models need few positively labelled query-documents pairs (qrels). Consequently, we evaluated them on 3 standard TREC collections :

- AP88-89 with topics 51-200 and 15 856 positive qrels
- FT91-94 with topics 251-450 and 6 486 positive qrels
- LA with topics 301-450 and 3 535 positive qrels

We use title of topics as queries. We lowercase, stem and remove stop-words from collections.

3.2 Baselines

We compare our models with several traditional IR models using a standard inverted index: TF-IDF, LM and BM25 and with neural supervised approaches for IR: DRMM [5], DUET [9] and Conv-KNRM [3]. DRMM performs matching based on a histogram of cosine similarities between word embeddings of query and document. The DUET model is a deep architecture that uses both local (exact matching signal) and distributed (word embeddings) representations to compute a relevance score. Conv-KNRM uses convolutions to generate several query-document interaction matrices that are processed by kernel pooling to produce learning-to-rank features.

3.3 Implementation

We implemented and trained our models with *Tensorflow* [1]. We used *MatchZoo* [6] for training and evaluation of neural baselines. We implemented traditional IR ranking functions in *Python*. We used word embeddings pre-trained on Wikipedia with the *fastText* [8] algorithm. Because of the limited amount of training data, we did not fine-tune word embeddings. In order to ensure that TDVs for all terms are non zero at the beginning of the training, we initialized bias b to 1. Weight vector w is initialized with the default *Tensorflow* initialization. To accelerate the training process, collection-level measures such as idf and collection language models were implemented batch wise and not collection wise. Preliminary experiments showed that dropout does not allow for better performance on the validations sets which is probably due to the low number of parameters of our models. Therefore, we do not use dropout in our experiments. 5-fold cross validation across the queries of the collections is used to tune hyperparameters. We use the Adam [7] algorithm to optimize models and early stopping on the training nDCG@5.

Method	AP88-89		LA		FT91-94	
	Base.	TDV	Base.	TDV	Base.	TDV
TF-IDF	147.4	29.2	26.3	4.6	56.6	15.1
LM	683.8	180.1	139.6	54.3	270.9	122.6
BM25	207.0	61.2	29.2	16.6	83.1	42.8
DRMM	$>10^4$	\	$>10^4$	\	$>10^4$	\
DUET	$>10^5$	\	$>10^5$	\	$>10^5$	\
Conv-KNRM	$>10^5$	\	$>10^5$	\	$>10^5$	\

Table 1: Comparison of the average retrieval time per query in milliseconds.

3.4 Evaluation

We assess three different evaluation measures: (1) standard IR metrics: nDCG@5 and Recall@1000; (2) inverted index's memory footprint reduction after removing terms with zero discrimination value; (3) average retrieval time per query. Statistically significant differences of nDCG@5 and Recall@1000 are computed with the two-tailed paired t-test with Bonferroni correction.

4 RESULTS

Retrieval speed up. Table (1) reports the retrieval time of the different approaches. First, we notice that the neural baselines are dramatically slower at retrieving documents than other models that use inverted indexes. Second, by filtering out terms with zero discrimination value from the inverted index, we are able to significantly speed up the retrieval process of all ranking functions on all collections. On LA and FT91-94, BM25 retrieval speed is almost doubled and on AP88-89 BM25 retrieval speed is tripled. Interestingly, we notice that LM consistently takes more time to retrieve documents than other models. Indeed, the ranking formula

Method	AP88-89				LA				FT91-94			
	nDCG@5		Recall@1000		nDCG@5		Recall@1000		nDCG@5		Recall@1000	
	Baseline	TDV	Baseline	TDV	Baseline	TDV	Baseline	TDV	Baseline	TDV	Baseline	TDV
TF-IDF	26.38	30.28*	53.08	58.42*	17.92	23.54*	60.05	65.09*	17.82	25.11*	50.98	55.55*
LM	44.64	46.30*	67.26	67.98	34.50	36.16	69.15	70.29	35.15	**37.78***	59.63	61.56
BM25	44.70	**47.09***	67.09	66.78	34.98	**40.04***	68.47	**72.00***	35.31	36.98*	60.40	**62.68**
DRMM	44.05	\	**68.24**	\	36.22	\	70.41	\	35.95	\	61.42	\
DUET	43.86	\	67.86	\	15.26	\	58.65	\	16.88	\	45.25	\
Conv-KNRM	44.13	\	67.36	\	22.65	\	60.41	\	37.95	\	51.42	\

Table 2: Performance comparison of the proposed models and baselines. Best results for each metric on each collection are highlighted in bold. * indicates a statistically significant improvement (p<0.05) of TDV over baselines. DRMM, DUET and Conv-KNRM do not outperform statistically significantly TDV-BM25 or TDV-LM.

Method	AP88-89	LA	FT91-94
TDV-TF-IDF	-45.00%	-39.67%	-45.77%
TDV-LM	-46.06%	-34.03%	-40.25%
TDV-BM25	-46.91%	-32.35%	-44.42%

Table 3: Inverted index memory footprint reduction.

of LM has to compute logarithms (that are computationally expensive) at retrieval time whereas BM25 and TF-IDF can compute such operations during indexation and use a lookup table at retrieval.

Effectiveness of proposed approaches. Table (2) shows the performance comparison of baselines and our models. Our main observation is that in most cases, learning and incorporating TDVs into IR ranking functions improves their performances despite the small amount of training data. Indeed, by construction and as a results of the bias initialization described in Section (3.3), our models performance are already close to the baselines at the start of the training process. Moreover, since we are in a limited data scenario, neural baselines perform poorly on the TREC collections compared to the traditional ones. The exception being DRMM as it has few parameters and does not require large amount of training data.

Memory footprint reduction. Table (3) describes the inverted index memory footprint reduction obtained when removing terms with zero discrimination value from the inverted index. For all collections, we are able to remove a significant portion of the inverted index and still outperform the original ranking functions.

5 CONCLUSION

In this paper, we proposed to learn TDVs using supervised learning. In order to have TDVs specific to traditional IR ranking functions and to be able to use neural networks, we developed a framework to make such functions differentiable and compatible with matrix operations. Moreover, our models can be trained with few positively labelled data. Removing terms with zero TDV at indexation leads to drastic retrieval speed up with a slight performance improvement compared to BM25 on several TREC collections. As future work, we are studying the correlation between the learned TDVs and count based formulae such as idf. We also plan to evaluate our models on collections with larger amount of labelled data.

REFERENCES

[1] M. Abadi, A. Agarwal, et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR* abs/1603.04467 (2016). arXiv:1603.04467 http://arxiv.org/abs/1603.04467

[2] F. Can and E. A. Ozkarahan. 1987. Computation of term/document discrimination values by use of the cover coefficient concept. *JASIS* 38, 3 (1987), 171–183.

[3] Z. Dai, C. Xiong, J. Callan, and Z. Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. 126–134.

[4] H. Fang, T. Tao, and C. Zhai. 2004. A formal study of information retrieval heuristics. In *SIGIR 2004: Sheffield, UK, July 25-29, 2004*. 49–56.

[5] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. 55–64.

[6] J. Guo, Y. Fan, X. Ji, and X. Cheng. 2019. MatchZoo: A Learning, Practicing, and Developing System for Neural Text Matching. In *SIGIR 2019, Paris, France, July 21-25, 2019*. 1297–1300.

[7] D. P. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

[8] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

[9] B. Mitra, F. Diaz, and N. Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *WWW 2017, Perth, Australia, April 3-7, 2017*. 1291–1299.

[10] C. Ramirez, V. Kreinovich, and M. Argaez. 2013. Why l1 is a good approximation to l0: A geometric explanation. *Journal of Uncertain Systems* 7, 3 (2013), 203–207.

[11] S. E. Robertson and H. Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[12] D. Roy, S. Bhatia, and M. Mitra. 2019. Selecting Discriminative Terms for Relevance Model. In *SIGIR 2019, Paris, France, July 21-25, 2019*. 1253–1256.

[13] G. Salton. 1975. *A theory of indexing*. Regional conference series in applied mathematics, Vol. 18. SIAM.

[14] G. Salton, A. Wong, and C.-S. Yang. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[15] H. Zamani, M. Dehghani, W. B. Croft, E. G. Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM 2018, Torino, Italy, October 22-26, 2018*. 497–506.

[16] C. Zhai and J. D. Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *SIGIR 2001: New Orleans, Louisiana, USA, September 9-13, 2001*. 334–342.