Attention based Term Weighting for App Retrieval

Lianzhi Tan Junjie Lin Shengping Zhou*
Platform and Content Group, Tencent
Hans Laser Technology Centre , Shennan Ave No.9988,
Nanshan District, Shenzhen City, Guangdong Province, 518057, China
{lianzhitan,alexjjlin,poissonzhou}@tencent.com

Abstract

App retrieval aims to select and rank relevant mobile apps to the textual queries users input, which is the key function for app search engines. As the terms in a query often play different roles in app retrieval, many term weighting schemes have been proposed to distinguish the importance of different terms in the query. However, most current term weighting schemes leverage only literal statistics information in texts, while neglecting user intentions implied in queries. Considering that the category of apps can reflect users' search intention to some extend, this paper proposes an attention based GRU-RNN network to learn the weight of each term by training on the data app category label. To the best of our knowledge, we are among the first to use attention-based deep networks to learn term weights for app retrieval. Experimental results show the effectiveness of our method, which achieves a 3.5% better nDCG@10 than baselines.

Introduction

App retrieval (also called app search) is a typical information retrieval task, which selects and ranks relevant apps from a large set of candidates according to the textual queries users input. Query understanding targets at identifying users' intention from searching more effective retrieval, thus can improve a user's search experience and boost a site's advertising profits. In the field of query understanding, term weighting is a fundamental task which aims to recognize important terms in retrieval. For example, users may input a query in an app retrieval engine: "download a game suitable for female playing." In order to achieve a better understanding of this query, the search engine needs to weight the input terms differently. Apparently, the weights of 'female' and 'game' should be larger than those of other terms. What's more, considering that the single term "game" may incur irrelevant apps (e.g., "competitive games"), the weight of "female" should be larger than that of "game". As shown in Figure 1, term weights indicate the importance of different terms in the query, and can be used for retrieving relevant terms.

In the literature, different term weighting schemes have been proposed in text retrieval, such as TF-IDF and its variant schemes (Beel, Langer, and Gipp 2017; Shirakawa, Hara, and Nishio 2015; G. Salton and Yang 1975; Chen et al. 2016; Soucy and Mineau 2005), part-of-speech tagging (Lioma and van Rijsbergen 2008; Lioma and Blanco 2017) and dependency parsing(Park and Croft 2010). Taking into account term weights in query, the related works propose many retrieval functions based on text similarity, such as, BM25 and TF-IDF based cosine similarity. BM25 (Svore and Burges 2009) is arguably one of the most widely used information retrieval functions. Structurally, BM25 and TF-IDF based functions are very similar (in the sense that they both consider term frequency and document frequency), however, they differ in many aspects. Firstly, BM25 is based on a well grounded theory, while the TF-IDF based schemes incline to an empirical background (Paik 2015). Secondly, unlike TF-IDF, BM25 uses a nonlinear query term frequency function (Paik 2015). Thirdly, BM25 is based on the standard IDF factor but it discounts the collection size by the document frequency of the term (Paik 2015). Approaches based on part-of-speech tagging and dependency parsing are limited to natural language queries, for which syntactic analysis of the query is feasible and reliable (Carmel et al. 2014a).

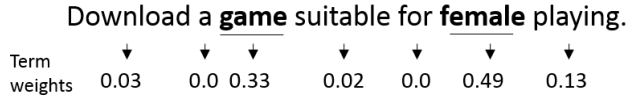

Figure 1: Example term weighting of query in app retrieval.

Beside these statistical linguistic analysis methods, term weighting essentially can be viewed as a classification problem (Qiu, Bao, and Shao 2013). Unfortunately, data labeling can be a huge project that costs human works. As app categories can reflect users' search intentions to some extent, and inspired the previous work of (Jin, Chai, and Si 2005) that learns term weights based on the correlation between word frequency and category information of documents, this paper takes app category as label and applies attention mechanism to learn term weights.

The contribution of this paper can be summarized as follows:

a).This paper is among the first to leverage a deep network for term weighting to improve the performance of app re-

*Corresponding author.

trieval. Although the attention based Bi-LSTM method is also proposed in the paper (Luo, Gong, and Chen 2018) for learning the central intention of user. The main goal of this method is inconsistent with ours, and the final performance on retrieval is unknown.
b).For app retrieval, the objectives of app category classification and term weighting is close. Therefore, this paper proposes the attention based GRU-RNN mechanism to learn term weights by training on the labeled data of app category classification. In case of lacking labeling of term weighting, our method has practical implications.
c).Finally, our method improve the performance of ranking metrics, such as nDCG@10, by 3.5% on app retrieval.

The remainder of this paper is organized as follows. Section II reviews the related works on term weighting schemes and app retrieval. Section III comprehensively introduces our method about term weighting based on attention mechanism. Section IV introduces data sets and experiments. Section V concludes this paper.

Related Works

2.1 Term Weighting Schemes for Information Retrieval

In most information retrieval tasks, a user's query is comprised of multiple terms. Distinct terms in the query contribute differently to retrieving relevant documents. Thus, term weighting schemes are commonly used in information retrieval models to assign an importance weight to each distinct term in the query.

TF-IDF model is one of the most widely used term weighting scheme. In TF-IDF model, the weight of a term is determined by the times it occurs in the query (i.e. term frequency, TF) as well as its inversed document frequency (i.e. IDF). A term is considered to be informative if it appears frequently in the query but appears rarely in the document collection. However, the largest weights tend to be assigned to high-frequency but less informative terms under this scheme. Besides, some work focuses on adjusting IDF to acquire more effective term weights (Rennie and Jaakkola 2005; Cooper, Chen, and Gey 1994; Shirakawa, Hara, and Nishio 2015). For example, (Rennie and Jaakkola 2005) propose Residual IDF measure, which assigns large weights to the terms whose IDF differs from the expected IDF significantly.

The above schemes mainly leverage literal information for term weighting. To integrate syntactic and semantic information into term weighting schemes, (Lioma and Blanco 2017) make use of part-of-speech (POS) n-grams to determine term weights. Based on the consideration that different POSs have distinct contributions to information retrieval, they propose five POS-based term weighting measures, two of which achieve relatively good performance, namely "pos_ml_weighted" and "pos_idf". "pos_ml_weighted" takes the frequencies of POS n-grams in the document collection to indicate how informative they are, and further uses the POS n-gram distribution of a term for term weighting. "pos_idf" adopts a simpler strategy which takes the inversed frequency of a term appearing in different POS n-grams as its weight. As these two measures consider only POS information in calculating term weights, they must be integrated into existing term weighting schemes for effective retrieval.

In addition, manually annotated query-document pairs are also used to predict important terms in query (Yih 2009). Other term weighting schemes consider issues such as term frequency heuristics (Buckley et al. 1995; Pasça 2001), users feedback (Saboori, Bashiri, and Oroumchian 2008), and the application of machine learning techniques (Cooper, Chen, and Gey 1994; Lioma and Blanco 2009; Bendersky and Croft 2008; Carmel et al. 2014b; Zheng and Callan 2015; Yih 2009), probabilistic models (Harter 1975; Margulis 1992) and graph-based algorithms (Arif, Rahman, and Mukta 2009) etc. Recently (Karisani, Rahgozar, and Oroumchian 2016) re-weight query terms according to their importance in the top retrieved documents by initial query.

2.2 App Retrieval

App retrieval (also called app search) (Cheng et al. 2016; Aliannejadi et al. 2018; Park et al. 2016) is the task of selecting and ranking relevant apps from a large set of candidates according to the textual queries users input, which can be regarded as a typical information retrieval problem. It should be noticed that app retrieval is a similar but different task from app recommendation (Yin et al. 2013; Lin et al. 2014) in terms of accepted input. App recommendation algorithms often take users behaviors and the information of apps as inputs, while app retrieval focuses on recognizing the most related apps to the input query string based on textual information.

Most of the existing methods for app retrieval recognize relevant apps by text similarity between query string and the descriptions of apps. Among various text similarity measures, TF-IDF based cosine similarity (Ramos and others 2003) and BM25 (Fang, Tao, and Zhai 2004) are both widely used and successfully applied in the field of information retrieval. To calculate TF-IDF based cosine similarity, queries and app descriptions are firstly represented as fixed-sized vectors, where each dimension corresponds to a specific term in the vocabulary and dimension values are TF-IDF term weights. Then the cosine similarity is calculated on the above vectors to indicate the relevance of query-app pairs. BM25 determines whether an app is relevant to the input query based on the terms occurring in both app description and query string. To take into account the influence of text length on similarity measurement, BM25 penalize term frequency in long app description. BM25 determines the importance of a term in retrieval by only its term frequency and document frequency.

Considering that the terms in a text is usually interrelated, some work incorporates language model for more effective retrieval (Zhai and Lafferty 2017; Ponte and Croft 1998). The Query Likelihood Language Model (QLLM) proposed by Ponte and Croft (Ponte and Croft 1998) scores the relevance of a query-document pair by the possibility that the terms in the query are generated from the document. The

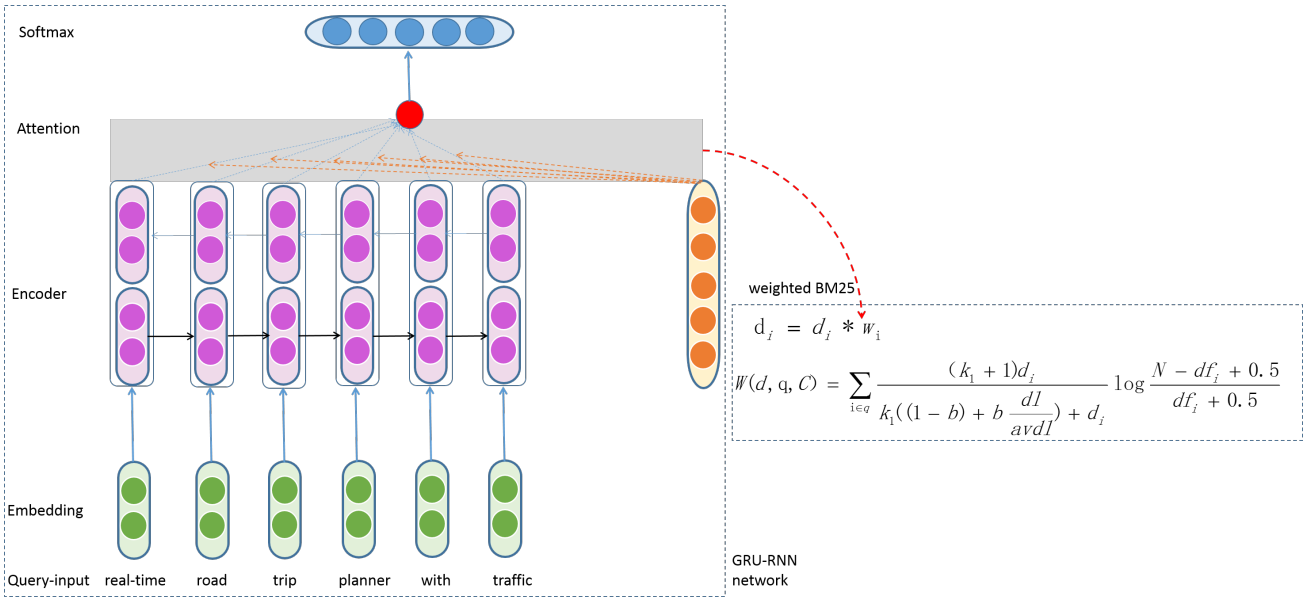

Figure 2: The proposed framework contains two parts: The GRU-RNN Network and The Weighted BM25.

latter estimator can prevent probability from being zero. To exploit more latent semantic information, topic models are leveraged to capture the topic relevance between query and document (Yi and Allan 2009; Li and McCallum 2006). For example, (Yi and Allan 2009) propose LDA-based document model (LBDM). On the basis of this, two works by (Park et al. 2016; 2015) make use of app reviews and users status texts in social media respectively to strengthen LBDM with more textual information for app retrieval.

In recent years, deep learning models have also been applied to app retrieval. (Aliannejadi et al. 2018) train a fully-connected feed-forward network to learn the relevance score of a query and an app. (Cheng et al. 2016) combine linear models with deep neural network to balance the memorization and generalization power of the retrieval model. In those works on app retrieval, few are related to term weighting. (Luo, Gong, and Chen 2018) propose a bi-LSTM based model to find the semantic relatedness between natural language context words and central intention term. However, this work only verifies the accuracy of term weighting, and the effect on retrieval is unknown.

Proposed Method

3.1 Overall Framework

In this paper, we proposes an attention based neural network for term weighting in app retrieval. The overall architecture of our proposed method is shown in Figure 2. Our method includes two parts, the attention based GRU-RNN network and BM25 with attention-based term weights. We will discuss them in 3.2 and 3.3 respectively. As shown in Figure 2, our proposed method works by the following steps:

Algorithm 1 Proposed Algorithm

1.Given app descriptions s_i containing T_i words, $i \in [1, L]$.
2.Train a attention based GRU-RNN model to predict o_i by equation 1-7.
3.For a query with Q_j words, calculate weights w_j by the word-level attention layer of the GRU-RNN network.
4.Rank the apps according to fusion methods of term weighting and term frequency.
5.Score the query against each app.

3.2 Attention based GRU-RNN for Term Weighting

The proposed attention based network for term-weighting consists of a embedding layer, a word sequence encoder, a word-level attention layer and a soft-max layer.

The embedding layer. Assume that descriptions of a app and a user's query are both word sequences s_i that contains T_i words. The embedding layer projects the raw app descriptions or queries into a vector representation and learns a hidden vector for each word. After the embedding layer, we get vectors from a sequence with words $w_{it}, t \in [0, T_i]$, through an embedding matrix $W_e, i.e. x_{ij} = W_e w_{ij}$.

The word sequence encoder layer. The word sequence encoder layer comprises of multiple GRU cells. GRU updates the state of a sequence by gating mechanism. At time t, the new state of GRU is computed as

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t. \quad (1)$$

where z_t represent the update gate which controls how information is updated to the new state. z_t is updated as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (2)$$

where x_t is the sequence vector at time t. The current candidate state $\widetilde{h}_t$ is computed according to the previous state h_{t-1} and current input x_t, which is computed with new sequence information. $\widetilde{h}_t$ represent the candidate state in equation 1, which is computed as

$$\widetilde{h}_t = tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h), \quad (3)$$

where r_t represent the reset gate, which decides how much the candidate state should remember the past state. Specifically, the previous state is forgotten completely when r_t is set to zero. r_t is updated as

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (4)$$

In the word sequence encoder layer, we get encoding representations of words by a bidirectional GRU (Bahdanau, Cho, and Bengio 2014) to summarize information for words in both directions. In the forward direction, a forward GRU $\overrightarrow{f}$ reads the sequence s_i from w_{i1} to w_{iT}. In the backward direction, a backward GRU $\overleftarrow{f}$ which reads from w_{iT} to w_{i1}. The bidirectional GRU is denoted as

$$\begin{aligned} x_{it} &= W_e w_{it}, t \in [1, T], \\ \overrightarrow{h}_{it} &= \overrightarrow{GRU}(x_{it}), t \in [1, T], \\ \overleftarrow{h}_{it} &= \overleftarrow{GRU}(x_{it}), t \in [T, 1]. \end{aligned} \quad (5)$$

where $\overrightarrow{h}_{it}$ represent the forward hidden state and $\overleftarrow{h}_{it}$ represent the backward hidden state. By concatenating the hidden state,i.e.,$h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$, we get summary of the whole sequence centered around w_{it}.

The word-level attention layer. The word-level attention layer provide weights of each word for classification. Considering that words of a sequence contribute differently to the meaning of the sequence, we use a word-level attention layer to learn weight terms that reflect the importance of each word in the sequence. The weighted sum of word representations are used to words forms a sequence vector, which is denoted as,

$$\begin{aligned} u_{it} &= tanh(W_w h_{it} + b_w) \\ \alpha_{it} &= \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)} \\ s_i &= \sum_t \alpha_{it} h_{it}. \end{aligned} \quad (6)$$

where u_{it} is a hidden representation of the word encoding representation h_{it}, α_{it} represents the normalized importance weight of each word. u_w is a word level context vector, which is initialized randomly and changed in procession of training. s_i indicates the sum of the weighted word encoding representation.

The softmax layer. The final softmax layer classify the sequence vector s_i to improve the performance of app category classification. The output probability o_i of s_i is

$$o_i = \frac{exp(s_i)}{\sum_i exp(s_i)} \quad (7)$$

3.3 BM25 with Attention-based Term Weights

We score a app against a query based on text similarity between query and app descriptions. We first define all the unstructured app descriptions as a collection C. Each text d is defined as a vector $(d_1, d_2, d_3, ...d_V)$ where d_i denotes term frequency of the ith term in d and V is the total number of terms in the vocabulary. The commonly used ranking functions BM25 defines a term weighting function $W(d, q, C)$ which exploits term frequency as well as other factors such as the app descriptions' length and collection statistics. For app retrieval, taking average term frequency into consideration, the standard BM25 function can be simplified to:

$$\begin{aligned} &W(d, q, C) = \\ &\sum_{i \in q} \frac{(k_1 + 1) d_i}{k_1((1 - b) + b\frac{dl}{avdl}) + d_i} log \frac{N - df_i + 0.5}{df_i + 0.5} \end{aligned} \quad (8)$$

where df_i is the frequency of term i in app description. $avdl$ is the average length of app descriptions across the collection, and k_1 and b are hyper-parameters.

$$d_i = d_i * w_i \quad (9)$$

The weight of each term w_i in the app description or query is acquired by the attention based GRU-RNN network. Our method re-weights d_i by multiplying it to w_i, which is denoted in equation 9.

Experiment

Datasets Carefully, we choose an app retrieval dataset that contains label of app category information, descriptions, and scores of query to top-10 app lists (Park et al. 2015). In Google Play app store, each app is assigned with one category only and there are 41 categories in total. Completely, there are 43,041 app descriptions (one description per app) are crawled, in which the average number of tokens is 94.1 and the total number of tokens is 4,051,366.

nDCG Measure The nDCG (Normalized Discounted Cummulative Gain) is a typical IR evaluation measure, which is based on DCG. DCG discountedly accumulates the scores of each recommendation result relevance as the score of the entire recommendation app lists. Given a app lists of k retrieval depth, the evaluation of DCG is

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i + 1)} \quad (10)$$

where rel_i is the relevance score of ith recommended app for query. The nDCG is computed over a set of queries by taking the arithmetic mean of the nDCG values of all the individual queries. The evaluation of nDCG is

$$nDCG@k = \frac{DCG@k}{IDCG@k} \quad (11)$$

where IDCG(Ideal DCG) means a list of ideal retrieval results, where the returned results are sorted by relevance, and the most relevant result is placed first.

query example	top-10 app names	app descriptions
text messenger my phone to my tablet	com.mango.findmyphone	if you lose your phone , just send it a text message and find my phone will reply to you with its current address , and a google maps link to your phone 's location.
	com.Socialize.PinkMessenger	please rate and comment socialize for facebook messenger features send and receive messages through your facebook account.
	com.tapixel.wireless.transferapp	send photo and video to another android phone tablet, just a few clicks and you are transferring files.
	com.lookout.labs.planb	it will use mobile-originated sms messages to send your location to your email address.
	com.apdroid.tabtalk	tablet talk is a tablet sms texting app that lets you text from a tablet and even make voice calls , while your phone stays in your pocket!
	biz.digitalshowcase.mycrapsgame.at142	tablets is almost exactly the same software as the ipad version.
	com.drippler.android.updates	love the new tablet app! drippler is like an android assistant or a user guide built specifically for my tablet
	com.zoemob.calltracker	zoemob family mobile tracker are sending or receiving bullying and sexting messages
	com.wyse.pocketcloudfree	This app is just brilliant. took me 3 minutes to install, connect and manage my remote windows 2008 server, without once referring to any of the documentation
	com.platinumapps.facedroid	you can also manage your facebook pages, groups, messages and chat with friends in real-time. and you can share new photos, a status update or check-in – all without launching the full application, so you'll enjoy faster sharing from your android-powered phone or tablet
best free internet based password saving app	com.np.comvigo	email address is your userid and password you will need email to login after installing after installing configure settings from your computer.
	com.miragestack.smart.phone.lock	If anyone tries to unlock your mobile with wrong password. The app is password protected , you can access the app only after entering the default pin which you have set during installation.
	com.nyxbull.nswallet	embedded password generator can be used to produce highly secure passwords.
	com.callpod.android_apps.keeper	Keeper's password generator creates high-strength passwords for your different websites , which is the best way to protect your privacy.
	com.reneph.passwordsafe	password safe is your solution! it stores and manages all entered data in an encrypted way , so you have a secure storage of your access data and only have to remind your master password.
	com.aed.droidvpn	Enter the email that you registered and the password that is sent to you
	com.passwordbox.passwordbox	Passwordbox is a free password manager that allows users to securely store , retrieve and share usernames , passwords and other personal data anytime.
	com.jagerinc.usagemonitor	A password is required to toggle this setting so only you will have access to it . remember to store the password created in a safe place.
	secureauth.android.token	You can start using your one-time password generator to gain access to your protected resources.
	com.osmino.wifil	You can also share the password to a known wi-fi network and make it available to other osmino wi-fi users.

Table 1: The top-10 apps of example queries got by the proposed model.

Experimental settings The initial learning rate of our proposed attention based GRU-RNN network is set to 0.001. The batch size is set to 32 and dropout rate is set to 0.8. Vocabulary size of queries is 5000. The embedding layer learns a 100-dimensional vector for each word. Sequence length of the word sequence encoder layer is 100. The word-level attention layer learns a 128 dimensional hidden representation. Number of epochs is set to 20 during the training process.

Comparison with Baseline Methods The experimental results are shown in Table 2. Our proposed weighting method outperforms the BM25 baseline method by 3.5% in terms of nDCG@10. We also compare the retrieval results calculated with the term weights initialized randomly. It can be seen that our method performs much better than BM25 with random weights.

Method	nDCG@10
Baseline(BM25)	74%
BM25+Random Weights	75%
Propose method	**77.5%**

Table 2: Experimental results of our model and baselines.

Comparison with State-of-the-art Methods We further investigate the effectiveness of our method compared with some state-of-the-art methods. As shown in Table 3. TF-IDF and POS tagging is proved to be effective in solving the problem of term weight in work (Chen et al. 2016) and (Lioma and Blanco 2017). In the experiment, we retest the method of TF-IDF and postagging which achieve 73% and 65% in nDCG@10 respectively. Experiment results show that the TF-IDF and postagging methods have the similar effect to the BM25 without term weighting. Our method weights terms by the attention layer of a deep network, learning the importance from app category labels, and thus improve the quality of the retrieved app list.

Method	nDCG@10
BM25	74%
TF-IDF	73%
POS-tagging	65.3%
Propose method	**77.5%**

Table 3: Experimental results of our method and state-of-the-art methods.

Case Study

In Table 1, we take query "text messenger my phone to my tablet" and "best free internet based password saving app" as examples to show some real cases of query result of top 10 app lists. For query "text messenger my phone to my tablet", the top-ranked app of the proposed model is "com.apdroid.tabtalk", whose app descriptions are "ablet talk is a tablet sms texting app that lets you text from a tablet and even make voice calls , while your phone, stays in your pocket!". We can see from Table 3 that the top-10 apps are all strongly relevant to user's query. We compare term weights of TF-IDF and our method on the example query shown in Figure 3. Apparently, the intention of the query is mainly related to "messenger", "tablet" and "phone". On the contrary, term of "my", "to" and "text" is comparatively not important. We calculate the nDCG@10 of this query and find that score acquired by our method is 0.99 while the score acquired by TF-IDF is 0.35. Another case is the query "best free internet based password saving app". The nDCG@10 of BM25 and TF-IDF is 0 and 0.62 respectively, while our proposed method improves this score to 0.75. However, there are some bad cases. For instance, the score of query "email custom folder" is 0 nDCG@10 in baseline, and turns no changing by our method. One possible reason for this may be that three terms in this query are not in the dictionary. Therefore, it's suggested that the larger is the corpus is, the better performance we will get.

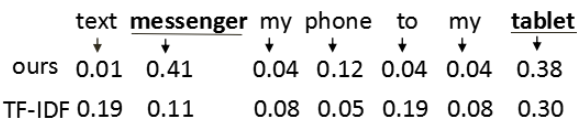

Figure 3: Compare the weights of TF-IDF and our method on the example query.

Conclusion

In this paper, we propose an attention based GRU-RNN network to learn term weights for app retrieval. In case of lacking labels to learn term weights, we propose to acquire term weights by training a neural network on the data with labels of app category classification. Specifically, we use the attention mechanism to learn the weights of words and integrate them to BM25, which is a typical score function in information retrieval. We finally use an app retrieval dataset for experimental study and demonstrate that our method outperforms baseline and the state-of-the-art methods. In addition, we analyze query examples from the experiment and verify that term weights acquired by our proposed method are more meaningful and reasonable. For future work, we shall apply this method to the online service of app retrieval data in our company.

References

Aliannejadi, M.; Zamani, H.; Crestani, F.; and Croft, W. B. 2018. Target apps selection: Towards a unified search framework for mobile devices. *arXiv preprint arXiv:1805.02211*.

Arif, A. S. M.; Rahman, M. M.; and Mukta, S. Y. 2009. Information retrieval by modified term weighting method using random walk model with query term position ranking. In *2009 International Conference on Signal Processing Systems*, 526–530.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Beel, J.; Langer, S.; and Gipp, B. 2017. TF-IDuF: a novel term-weighting scheme for user modeling based on users personal document collections. *Proceedings of the 12th iConference.*

Bendersky, M., and Croft, W. B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 491–498.

Buckley, C.; Singhal, A.; Mitra, M.; and Salton, G. 1995. New retrieval approaches using SMART: TREC 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 25–48.

Carmel, D.; Mejer, A.; Pinter, Y.; and Szpektor, I. 2014a. Improving term weighting for community question answering search using syntactic analysis. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 351–360.

Carmel, D.; Mejer, A.; Pinter, Y.; and Szpektor, I. 2014b. Improving term weighting for community question answering search using syntactic analysis. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 351–360.

Chen, K.; Zhang, Z.; Long, J.; and Zhang, H. 2016. Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications* 66:245–260.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10.

Cooper, W. S.; Chen, A.; and Gey, F. C. 1994. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. *NIST SPECIAL PUBLICATION SP* 57–57.

Fang, H.; Tao, T.; and Zhai, C. 2004. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 49–56.

G. Salton, A. W., and Yang, C.-S. 1975. A vector space model for automatic indexing. 613–620.

Harter, S. P. 1975. A probabilistic approach to automatic keyword indexing. part i. on the distribution of specialty words in a technical literature. *Journal of the american society for information science* 26(4):197–206.

Jin, R.; Chai, J. Y.; and Si, L. 2005. Learn to weight terms in information retrieval using category information. In *Proceedings of the 22nd international conference on Machine learning*, 353–360.

Karisani, P.; Rahgozar, M.; and Oroumchian, F. 2016. A query term re-weighting approach using document similarity. *Information Processing & Management* 52(3):478–489.

Li, W., and McCallum, A. 2006. Pachinko allocation: DAG-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, 577–584.

Lin, J.; Sugiyama, K.; Kan, M.-Y.; and Chua, T.-S. 2014. New and improved: modeling versions to improve app recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 647–656.

Lioma, C., and Blanco, R. 2009. Part of speech based term weighting for information retrieval. In *European Conference on Information Retrieval*, 412–423.

Lioma, C., and Blanco, R. 2017. Part of speech based term weighting for information retrieval. *arXiv preprint arXiv:1704.01617*.

Lioma, C., and van Rijsbergen, C. K. 2008. Part of speech n-grams and information retrieval. *Revue française de linguistique appliquée* 13(1):9–22.

Luo, X.; Gong, Y.; and Chen, X. 2018. Central intention identification for natural language search query in e-commerce.

Margulis, E. L. 1992. N-poisson document modelling. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 177–189.

Paik, J. H. 2015. A probabilistic model for information retrieval based on maximum value distribution. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 585–594.

Park, J. H., and Croft, W. B. 2010. Query term ranking based on dependency parsing of verbose queries. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 829–830.

Park, D. H.; Liu, M.; Zhai, C.; and Wang, H. 2015. Leveraging user reviews to improve accuracy for mobile app retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 533–542.

Park, D. H.; Fang, Y.; Liu, M.; and Zhai, C. 2016. Mobile app retrieval for social media users via inference of implicit intent in social media text. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 959–968.

Pasça, A. M. 2001. *High-performance, open-domain question answering from large text collections*. Southern Methodist University.

Ponte, J. M., and Croft, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 275–281.

Qiu, Y.; Bao, L.; and Shao, L. 2013. Term importance identification method based on classification. *Computer Science* 40(11):242–247.

Ramos, J., et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, 133–142.

Rennie, J. D., and Jaakkola, T. 2005. Using term informativeness for named entity detection. In *Proceedings of*

the 28th annual international ACM SIGIR conference on Research and development in information retrieval, 353–360.

Saboori, F.; Bashiri, H.; and Oroumchian, F. 2008. Assessment of query reweighing, by rocchio method in farsi information retrieval.

Shirakawa, M.; Hara, T.; and Nishio, S. 2015. N-gram idf: A global term weighting scheme based on information distance. In *Proceedings of the 24th International Conference on World Wide Web*, 960–970.

Soucy, P., and Mineau, G. W. 2005. Beyond TFIDF weighting for text categorization in the vector space model. In *Proceedings of the 19th international joint conference on Artificial intelligence*, volume 5, 1130–1135.

Svore, K. M., and Burges, C. J. 2009. A machine learning approach for improved BM25 retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 1811–1814.

Yi, X., and Allan, J. 2009. A comparative study of utilizing topic models for information retrieval. In *European conference on information retrieval*, 29–41.

Yih, W.-t. 2009. Learning term-weighting functions for similarity measures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, 793–802.

Yin, P.; Luo, P.; Lee, W.-C.; and Wang, M. 2013. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, 395–404.

Zhai, C., and Lafferty, J. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, 268–276.

Zheng, G., and Callan, J. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 575–584.